UNIVERSIDADE FEDERAL DE MINAS GERAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA

Uma contribuição metodológica ao projeto, modelagem matemática e ao controle baseado em inteligência artificial de uma cadeira de rodas servoacionada

CARLOS ANTONIO RENNÓ

Belo Horizonte, 21 de Fevereiro de 2014

UNIVERSIDADE FEDERAL DE MINAS GERAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA

Carlos Antonio Rennó

Uma contribuição metodológica ao projeto, modelagem matemática e ao controle baseado em inteligência artificial de uma cadeira de rodas servoacionada

Tese apresentada ao Programa de Pós-Graduação em Engenharia Mecânica da Universidade Federal de Minas Gerais, como requisito parcial à obtenção do título de Doutor em Engenharia Mecânica.

Área de concentração: Projetos Mecânicos

Orientador: Prof. Ricardo Poley Martins Ferreira - Universidade Federal de Minas Gerais; Escola de Engenharia da UFMG

Coorientador: Prof. Eduardo José Lima II - Universidade Federal de Minas Gerais; Escola de Engenharia da UFMG

Belo Horizonte

Escola de Engenharia da UFMG

2014

UNIVERSIDADE FEDERAL DE MINAS GERAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA

Uma contribuição metodológica ao projeto, modelagem matemática e ao controle baseado em inteligência artificial de uma cadeira de rodas servoacionada

CARLOS ANTONIO RENNÓ

Tese defendida e aprovada em 21 de Fevereiro de 2014 pela Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Mecânica da Universidade Federal de Minas Gerais, como parte dos requisitos necessários à obtenção do título de "**Doutor em Engenharia Mecânica**", na área de concentração de "**Projetos Mecânicos**".

____

Prof. Dr. Ricardo Poley Martins Ferreira – UFMG – Orientador

____

Prof. Dr. Eduardo José Lima II – UFMG – Coorientador

____

Prof. Dr. Ricardo Luiz Utsch de Freitas Pinto – UFMG – 1º. Examinador Interno

____

Prof. Dr. Guilherme de Souza Papini – UFMG – 2º. Examinador Interno

____

Prof. Dr. Ronan Drummond de Figueiredo Rossi – CEFET/MG – Centro Federal de Educação Tecnológica – 1º. Examinador Externo

____

Prof. Dr. Enguer Beraldo Garcia –FCM/MG – Faculdade de Ciencias Médicas – 2º. Examinador Externo

Dedico este trabalho aos meus pais, Olavo e Regina (1929-1999), à minha primeira esposa, Marilene Leopoldino Rennó (1952-1995), já falecidos, quando tudo começou; ao George Lucas Rennó, filho deste meu primeiro casamento; à Janete Vieira Rennó, minha atual esposa; ao Gustavo Vieira Rennó, filho deste meu segundo casamento, que reacendeu meu ser e que tanto me apoiaram ao longo dos meus estudos; a toda minha família e amigos sinceros, que me ajudaram nesta trajetória tortuosa e continuamente insegura do viver, trazendo a luz da razão na penumbra de nossas vidas, outrora mergulhadas na escuridão do não saber.

AGRADECIMENTOS

Aos orientadores que me apoiaram ao longo deste desenvolvimento, professores Eduardo José Lima II, que consolidou a revisão da tese e Ricardo Poley Martins Ferreira, dotados de uma inteligência muito acima da média e com a qual contei para as correções deste trabalho e artigos feitos a seis mãos, agradeço o auxílio fundamental na análise da tese, um estudo baseado em inteligência artificial. Minha gratidão a eles será para sempre.

Agradeço, com igual penhor, aos professores Ricardo Luiz Utsch de Freitas Pinto e ao seu ex-orientado, Guilherme Papini, a excelência de professores da UFMG que são, a amizade, o interesse, a disponibilidade absoluta, as críticas construtivas e o suporte despretensioso durante o desenvolvimento de todas as fases deste trabalho. Gostaria de deixar registrado aqui a minha gratidão a estes singulares professores de reconhecida competência.

Agradeço igualmente ao professor Teodiano Freitas Bastos Filho, e me lembro claramente do dia em que o professor sugeriu este trabalho, para que juntos pudéssemos apresentá-lo no Iberdiscap 2000, realizado na cidade de Madrid, Espanha. Depois disso, apresentei vários outros no MWSCAS2000, realizado na cidade de Lancey, Estado de Michigam, USA, CONEM2012, em São Luís do Maranhão, e IEEE-Bios Signals2013, na cidade do Rio de Janeiro. O professor Teodiano foi quem me incentivou, em minha breve passagem pela UFES, Vitória (ES), a lançar esta semente que após tantos anos floresceu.

Ao professor Alexandre Queirós Bracarense, agradeço o suporte e apoio ao longo desses anos na UFMG e no LRSS – Laboratório de Robótica, Soldagem e Simulação.

Aos meus colegas, demais professores e funcionários não só da UFMG como também do CEFETMG, agradeço o companheirismo e incentivo durante minha permanência nesse processo.

Ao Eng. Jorge Kiefer**,** do CEFET/MG, amigo leal, participante do desenvolvimento inicial deste experimento, agradeço-lhe o apoio e amizade absolutos ao longo de vários anos.

A você, leitor, para que possa aproveitar este trabalho aberto ao público, melhorando-o com críticas e sugestões.

Os meus mais sinceros agradecimentos.

“Se você não tiver certeza que vai terminar uma tarefa, não a comece. Uma vez iniciado, termine-a...”.

(Provérbio chinês)

“Não é porque as coisas são difíceis que nós não ousamos. É justamente porque não ousamos que as coisas são difíceis”.

(Sêneca-filósofo romano.65 aC a 4aC)

“De que vale ao homem conquistar toda a Terra se depois perder a sua alma”.

(Mateus - discípulo cristão - escritor de um dos evangelhos do novo testamento)

“Um pessimista vê uma dificuldade em cada oportunidade; um otimista vê uma oportunidade em cada dificuldade.”

(Winston Churchill)

“É permissível a cada um de nós morrer pela sua fé, mas não matar por ela.” (Hermann Hesse)

“Existem muito mais coisas entre o Céu e a Terra do que sonha a nossa vã filosofia.” (William Shakespeare)

“Cur non facere serius exhilaratus”

Por que não fazer o sério de forma divertida?

Revisão e Normalização realizados por:

LEANDRO NEGREIROS CPF 053.541.856-61 – Belo Hte – MG - Brasil

SUMÁRIO

Revisão e Normalização realizados por:

LEANDRO NEGREIROS CPF 053.541.856-61 – Belo Hte – MG - Brasil

LISTA DE FIGURAS

LISTA DE TABELAS

Revisão e Normalização realizados por:

LEANDRO NEGREIROS CPF 053.541.856-61 – Belo Hte – MG - Brasil

LISTA DE ABREVIATURAS E NOMENCLATURAS

A Matriz da parte linear do modelo sem escorregamento da cadeira.
A Amplitude do sinal.
$\mathbf{A}_d$ Matriz do modelo discreto obtido por discretização do modelo sem escorregamento.
A/D Conversão Analógico/Digital
ANN "Artificial Neural Network" (Rede Neural Artificial).
ART "Adaptive Resonance Theory"
$\boldsymbol{\Lambda}$ Matriz diagonal de autovalores de *R*
a Aceleração linear da cadeira robótica (d*v*/dt).
A Aceleração angular da cadeira (d*ω*/dt).
α_β Aceleração angular da cadeira robótica no plano ($d^2\beta/dt^2$).
α_θ Aceleração angular da cadeira fora do plano ($d^2\theta/dt^2$).

B Matriz da parte linear do modelo sem escorregamento da cadeira.
$\mathbf{B}_d$ Matriz do modelo discreto obtido por discretização do modelo sem escorregamento.
B Distância entre o ponto ***O*** e o ponto médio nas rodas ou coeficiente de atrito dinâmico.
β Ângulo [rad]
β Ângulo de orientação da cadeira (medido em relação ao eixo xx' no sentido do plano).
β_p Ângulo da tangente ao percurso no ponto mais próximo deste da cadeira robótica.
β_{ref} Referência de orientação da cadeira no plano.

C
$\boldsymbol{C_e}$ Capacitância elétrica [Faraday]
c_1/c_8 Constantes do modelo da cadeira robótica sem atrito.
C Coeficiente de atrito estático (Coulomb).

D Diâmetro

DSP Processamento digital de sinais (Digital Signal Processor)

d(n) Resposta desejada no tempo discreto n

d Distância entre o ponto ***O*** e o CG.

d_1/d_5 Constantes do modelo da cadeira robótica com atrito.

$\partial\xi(n)/\partial w_l$ Derivada direcional

Δ_θ Ângulo entre retas de aproximação no método de seguimento de referências variáveis.

Δ_{max} Ângulo máximo entre retas de aproximação no método de seguimento de referências variáveis.

E Tensão atuante [V]

E[ ] Valor esperado

E_a Tensão de armadura do motor de corrente contínua.

E{x(t)} Valor "médio" do valor esperado de x(t), i.e. limN-+- ...!..fE{x(I)} N , •• 1

Eq Sigla para equações

e(n) Erro absoluto

$e^2(n)$ Erro quadrático instantâneo

e Vetor erro.

e_p Erro de distância. Distância entre a posição atual e o percurso de referência.

e_β Erro de orientação. Diferença angular entre a orientação atual e a da referência ou erro de orientação no método de seguimento de caminhos (diferença entre orientação atual e tangente à trajetória no seu ponto mais próximo).

e_v Erro de velocidade linear.

e_ω Erro de velocidade angular.

e_θ Erro de orientação no método de referência virtual.

e_d Erro de posição, distância entre a posição atual e a posição de referência (ou alvo).

EMQ Erro Médio Quadrado

ξ_{min} Erro médio quadrático mínimo

$\xi(n) \equiv E[e^2(n)]$ - Erro médio quadrático

F/F$_x$ Força ou até mesmo a ação de impulsão da cadeira robótica (Consequente de Torque).

F_d Força gerada na roda motriz direita.

F_e Força gerada na roda motriz esquerda.

F_{sis} Força mecânica atuando no sistema [N]

f_a Força de atrito estático.

f_e Fator de escorregamento, usado para medir o escorregamento global da cadeira.

f_s Freqüência de amostragem [Hz]

f_v Força de atrito viscoso.
ϕ Ângulo [rad]

G/G$_x$ Força centrípeta.
GHz Gigahertz (1000000 x Hz)
g Valor da aceleração da gravidade (≈ 9,8 m/s^2).
γ Ângulo [rad]
$\nabla\xi(\boldsymbol{n})$ Gradiente da função erro em relação aos coeficientes

H Altura do CG ao plano do assento da cadeira.
HMM Hidden Markov Models em redes neurais
Hz Hertz – Unidade básica de frequência
H(z) Função de transferência
h Distância entre o ponto de apoio de uma roda livre e o ponto ***O*** ou avanço do alvo de referência ou intervalo (de tempo ou espaço) entre pontos de comutação no método de seguimento de referências variáveis.
h_r Distância entre retas aproximativas no método de seguimento de referências variáveis.

I Inércia
$\boldsymbol{I_t}$ Corrente total do circuito elétrico [A]
$\boldsymbol{i}$ Corrente elétrica [A]

J Momento de inércia também definido como ***I***.
J_O Momento de inércia da cadeira em torno do eixo vertical que passa pelo ponto O.
J_w Momento de inércia das rodas motrizes em torno do seu eixo de rotação.

K
KHz Kilohertz (1000 x Hz)
k Constante
k1, k2 Constantes elétricas do motor direito.
k1/,k2/ Constantes elétricas do motor esquerdo.
$\boldsymbol{k_{mola}}$ Constante da mola [N/m]
k_T Constante de Torque do motor (genericamente dir. e esq.).

L Largura das rodas motrizes ou indutância dos motores.
L Comprimento de peça ou haste [m]
$\boldsymbol{L_e}$ Indutância elétrica [H]

LQG	Regulador linear quadrático gaussiano.
LQR	Regulador linear quadrático ("Linear Quadratic Regulator").
LTR	Recuperação de ganho de malha ("Loop Transfer Recovery").
LPC	Coeficientes de Predição Linear
LMS	Least Mean-Square
$\boldsymbol{\lambda}$	Comprimento de onda [m]
λ	Usada como constante de proporcionalidade representando a taxa de aprendizado em redes neurais.
λ_1	Escorregamento ("slip"), medida do escorregamento da roda 1.
λ_2	Escorregamento ("slip"), medida do escorregamento da roda 2.
M	Massa total da cadeira inclusive usuário.
M	Massa maior [kg]
MIMO	"Multiple-Input Multiple-Output" (Sistema múltipla entrada, múltipla saída).
MC&P	McCulloch & Pitts – Nomes citados no desenvolvimento do Neurônio Artificial.
MLP	"Multi-Layer Perceptron" – Rede neural Perceptron de várias camadas.
MSVQ	"Multi-Section Vectorial Quntization" (Quantização Vetorial Multiseção).
m	Massa menor [kg]
m	Massa total da cadeira sem usuário.
ms	Milisegundos (s/1000).
mV	Milivolts (V/1000).
$\boldsymbol{\mu}$	Fator de convergência em redes neurais
μ	Coeficiente de atrito.
μ_d	Coeficiente de atrito dinâmico também denominado de ***b***.
μ_s	Coeficiente de atrito estático também denominado de ***f***.
μ_{d1}	Função que descreve o coeficiente de atrito da roda 1 (determinada empiricamente).
μ_{d2}	Função que descreve o coeficiente de atrito da roda 2 (determinada empiricamente).
μ_L	Função que descreve o coeficiente de atrito lateral (determinada experimentalmente).
N	Indicativo de ordem ou força Normal
n	Numero de Ordem
ns	Nanosegundos (s/1000000).
$\mathbf{O(n^x)}$	Custo de processamento dentro de determinado método no qual **n** é o numero de vezes que se processa **x**, o termo mais "pesado em potência".
O	Ponto de referência inercial local situado no meio do eixo imaginário traseiro da cadeira robótica.

P	Distância entre ponto de comutação de referência e reta de aproximação no método de seguimento de referências variáveis.
PC	Personal Computer® IBM ®
PID	Controle Proporcional-Integral-Derivativo.
$\boldsymbol{P_e}$	Potência elétrica do motor [kW]
$\boldsymbol{p}$	Pressão [Pa]
Q	Matriz de peso do estado na função de objetivo do LQR.
$\boldsymbol{Q}$	Matriz de autovetores
R	Matriz de peso do controle na função objetivo do LQR.
$\boldsymbol{R}$	Matriz de correlação
RNAs	Redes Neurais Artificiais.
RAV	Reconhecimento Automático de Voz.
RN	Rede Neural.
R_a	Resistência interna da armadura do motor (genericamente dir. e esq.).
R_c	Raio instantâneo de curvatura.
$\boldsymbol{R_e}$	Resistência elétrica [ohm]
$\boldsymbol{R_c}$	Resistência interna da carga [ohm]
$\boldsymbol{R_i}$	Resistência interna do indutor [ohm]
$\Re$	Parte real ou valores reais.
r	Raio das rodas motrizes.
S	
$\boldsymbol{S}$	Sensores ou Área da seção transversal [m^2]
SISO	Sistema de uma entrada e uma saída ("Single-Input Single-Output").
s	Segundo – Unidade básica de tempo.
T	Período ou tempo
$\boldsymbol{T_{freq}}$	Período da frequência a ser tratada [s]
$\boldsymbol{T_{op}}$	Temperatura de operação [^{0}C]
$\boldsymbol{T_h}$	Temperatura ambiente [^{0}C]
t_i	Instante de tempo para o qual começa um dado inicial da trajetória ("offset").
t_p	Instante de tempo correspondente à posição mais próxima da referência
tr	Traço de uma matriz
θ	Ângulo [rad]

$\boldsymbol{U}$ Ação de controle. Costuma ser a própria Força F atuante.
$\boldsymbol{U}_d$ Controle (referência) para o motor direito.
$\boldsymbol{U}_e$ Controle (referência) para o motor esquerdo.
u Vetor de controle.
u/ Valor nominal dos dois controles.
u_1/ Valor nominal de u1.
u_2/ Valor nominal de u2.

V

$\boldsymbol{V}$ Volume do sistema [m^3]
$\boldsymbol{V_{max}}$ Velocidade máxima [m/s]
V Volt - Unidade básica de tensão elétrica.
V_a Tensão aplicada ao motor de corrente contínua.
$\boldsymbol{v}$ Velocidade linear da cadeira robótica.
$\boldsymbol{v}_d$ Velocidade linear da roda motriz direita.
$\boldsymbol{v}_e$ Velocidade linear da roda motriz esquerda.

X

$\vec{x}(n)$ Vetor do sinal de referência
x Vetor de estado. Pode ser um vetor de entrada em Redes Neurais.
x_i Valor de um parâmetro de entrada em redes neurais.
x Coordenada x da posição da cadeira robótica (ponto ***O***).
x_p Coordenada x da posição na referência mais próxima da posição atual da cadeira.
x_{alvo} Coordenada x da posição do alvo de referência.
x_{ref} Coordenada x da referência em posição.

Y

y(n) Sinal de saída de um filtro digital no tempo discreto n
y Vetor de saída em Redes Neurais.
y_j Valor de saída em Redes Neurais.
y Coordenada y da posição da cadeira robótica (ponto ***O***).
y_p Coordenada y da posição na referência mais próxima da posição atual da cadeira.
y_{alvo} Coordenada y da posição do alvo de referência.
y_{ref} Coordenada y da referencia em posição.
ζ Variável de estado usada nos modelos para as rodas livres ou fator de descrição da resposta de um controlador para retas, no método de seguimento de referências variáveis.

$\boldsymbol{Z}$	Impedância [Ohm]
Z_t	Impedância total do circuito [Ohm]
$\boldsymbol{Z_0}$	Impedância característica [Ohm]

W

$\boldsymbol{W_m}$	Potência mecânica [cv]
$\boldsymbol{W_e}$	Potência elétrica [W]
WMR	Robô móvel terrestre com rodas ("Wheeled Mobile Robot").
WTA	Winner Take All ou o ganhador leva tudo nas redes neurais
$\boldsymbol{\vec{w}^0}$	Solução ótima para $\vec{w}(n)$
$\boldsymbol{\omega}$	Velocidade angular [rad/s]
$\boldsymbol{\omega}_0$	Velocidade angular inicial [rad/s]
ω_θ	Velocidade angular da cadeira fora do plano (dθ/dt).
ω_β	Velocidade angular da cadeira robótica no plano (dβ/dt).
Ω	Ângulo [rad]

RESUMO

Existem diversos dispositivos projetados pelo ser humano que apresentam elevadas características não lineares, inerentemente instáveis e variáveis no tempo. Esses dispositivos são difíceis de serem modelados e controlados. Os métodos clássicos que são usados para sistemas lineares apresentam limitações quando aplicados a sistemas com essas características. Alternativas modernas abordam o estudo, modelo e controle de tais sistemas, mas algumas são denominadas de não convencionais, mais conhecidas como baseadas em conhecimento e incluem o uso de ferramentas de inteligência artificial. Este trabalho descreve o desenvolvimento da modelagem e controle de um sistema com as características citadas, baseado em um controle não convencional, com o emprego da Lógica “Fuzzy”. Esta, devido ao seu baixo custo (de projeto e processamento), situa-se em um patamar que permite a um leigo (fisioterapeutas, técnicos ortopédicos, médicos bio–engenheiros e outros) em relação ao controle convencional projetar e executar um projeto, desde que tenha conhecimento do seu funcionamento. O presente estudo investigou essas novas técnicas aplicadas a um caso real, de uma cadeira de rodas para desabilitados físicos tetraplégicos com a opção da inclinação. Apresentaram-se, aqui, adicionalmente, dois estudos simples, mas que foram fundamentais no controle por parte do tetraplégico, como o comando vocal baseado em Redes Neurais Artificiais e outro, que traz certa inovação para o caso do Centro de Gravidade (CG) de uma cadeira de rodas calculado em tempo real e que o diferencia das técnicas usuais empregadas na estabilização de dispositivos como este.

Palavras-Chave: Cadeira de Rodas; Modelo Matemático; Controle Neuro-Fuzzy; Robotica.

ABSTRACT

There are many devices designed by humans, which have high nonlinear characteristics that are inherently unstable and variable in time. These devices are difficult to be modeled and controlled. Classics methods that are used for linear systems have limitations when applied to systems with these characteristics. There are modern alternative approaches to studying, modeling and control such systems, but some are called unconventional, better known as knowledge-based and include the use of artificial intelligence tools. This paper describes the development of a model and control system with the aforementioned characteristics, based on a non - conventional control, with the use of fuzzy logic. Fuzzy logic and its inherently low cust (designing and processing) is situated on a level that allows a layman (physiotherapists, orthopedic technicians, doctors, bio - engineers, etc.) to design and execute a project, since it is aware of its usual way. This work investigates a new technic applied in a real case of a quadriplegic's wheelchair with tilt option. We present here additionally two simple studies, but that were fundamental in control by the quadriplegic as voice control based on artificial neural networks and another that brings some innovation to the case of the Center of Gravity (CG) of a wheelchair, computed in real time and which differentiates it from the usual techniques in stabilizing devices like this.

Keyword: Wheelchair; Mathematical Model; Neuro-Fuzzy Control; Robotic.

Revisão e Normalização realizados por:

LEANDRO NEGREIROS CPF 053.541.856-61 – Belo Hte – MG - Brasil

1 INTRODUÇÃO

1.1 Visão geral do problema

Por mais que se tenham desenvolvido artigos e projetos de cadeira de rodas para os deficientes, alguns de admirável fundamentação científica, estas ainda se encontram em uma situação limitada de movimentos e de apoio da robótica aos seus usuários, entre eles os tetraplégicos.

Apesar de vários projetos de cadeiras terem falhado comercialmente, esta triste constatação (LOMBARDI, 2002), realizada através das revisões bibliográficas, apontou uma nova linha de pesquisa científica que bem poderia ser a espinha dorsal desta Tese. Assim, este trabalho convergiria para um estudo da dinâmica de uma cadeira tanto sob o aspecto convencional como não convencional (com o apoio de um controle baseado em inteligência Neuro-Fuzzy), de forma comparativa, sendo que o controle final resultante seria efetivamente implementado sob uma forma mais simples e econômica, em uma estratégia baseada em algoritmos de Inteligência Artificial.

A utilização da Inteligência Artificial no reconhecimento de voz e controle no trabalho que se apresenta ignora, de certa forma, mas coerentemente, as precauções tomadas pelo controle convencional com os aspectos da não linearidade do sistema a ser desenvolvido. O objetivo disso é mostrar tal como um recurso matemático de aproximação complexo e eficiente, eliminando-se, neste caso, a aproximação do sistema sob vários blocos de controle realimentados, ao se utilizar a teoria de controle convencional clássico, que pode ser uma prática perigosa, levando o sistema a um controle linear instável, como no caso do controle PID.

De acordo com estimativas atuais da Organização Mundial de Saúde (OMS) (CHAN, 2014), há no mundo de dez a doze por cento da população mundial (algo em torno de 700 a 800 milhões de pessoas) com alguma deficiência física. Destes indivíduos, perto de 90% vivem nos chamados países em desenvolvimento, e o mesmo percentual vale para os que estão em idade produtiva, das quais cerca de 63% fazem parte da população economicamente ativa, mas vivem desempregados. Esse conceito de deficiência inclui uma grande variedade de características físicas, intelectuais ou sensoriais decorrentes de acidentes, de doenças adquiridas ou genéticas.

No Brasil, o Instituto Brasileiro de Geografia e Estatística (IBGE) divulgou, em 2012, os resultados do censo realizado em 2010 e verificou a existência de 23,9% da população, ou 45,6 milhões de pessoas, com deficiência. A maior parte desse contigente vive em áreas urbanas, 38.473.702, ante 4.132.347, nas áreas rurais. Os dados foram levantados segundo os critérios previstos na Classificação Internacional de Funcionalidade, Incapacidade e Saúde (CIF), conforme recomendação da OMS. Desse total, aproximadamente 42% (ou 10,2 milhões) possuem deficiências severas, tais como deficiência mental permanente, tetraplegia, paraplegia ou hemiplegia permanente, falta de um membro ou de parte dele e incapacidade ou grande dificuldade permanente de caminhar, subir escadas, enxergar e ouvir.

Adaptações indispensáveis foram criadas em todas as partes do mundo para tornar as cadeiras de rodas ágeis e seguras para o uso em determinados esportes, tais como as corridas e maratonas, o basquetebol e o tênis em cadeiras de rodas. Há modelos surpreendentes nesse campo, muito leves, com eixos especiais e muito maior proximidade do solo. Mas 75% desses projetos são para acionamento manual, o que não é o interesse deste trabalho, voltado ao acionamento elétrico.

Um trabalho observado inicialmente, para a elaboração do projeto que se apresenta, foi o desenvolvido por uma firma americana, sob a direção do Sr. Dean Kamen (um autodidata e prático,) a cadeira iBot®, adquirida pela Johnson&Johson®, com mais graus de liberdade, que, além da navegação no plano horizontal, inclue movimentos fora deste plano, apontando para diversas outras estratégias de projeto. Além disso, provou-se, nos primeiros testes, ser viável sob o conceito da engenharia, mas com resultados essencialmente limitados na área comercial (Custo elevado entre US$16.000 e US$22.000) (DEYLE, 2009).

Muitas campanhas têm sido desenvolvidas em várias partes do mundo, procurando chamar a atenção para os aspectos de acessibilidade que afetam sobremaneira as pessoas que usam cadeiras de rodas. Através da Internet, estão levando o acesso a todos os ambientes. A Rehabilitation International aprovou seu projeto de ícone indicativo de acesso a cadeiras de rodas (aprovado pelo Sistema ISO de qualidade), que se tornou internacional. Outros esforços têm sido desenvolvidos continuamente e precisam ser sempre apoiados.

Durante séculos, as pessoas com deficiência foram marginalizadas pela sociedade, isoladas em instituições ou em suas próprias casas. Somente na década de 60, é que se iniciaram os movimentos reivindicatórios, organizados pelas pessoas com deficiência que passaram a lutar por seus direitos. As conquistas foram pouco a pouco se transformando em leis e hoje tanto a legislação nacional quanto a internacional incluem a garantia de acesso ao trabalho. Embora os sistemas utilizados no Brasil não prevejam a aplicação de adequações por

parte das instituições, a progressiva conscientização por parte da sociedade, aliada à ação fiscalizadora do Ministério Público, tem estimulado a adequação (das empresas) à legislação vigente.

Atualmente, tem-se verificado uma presença crescente de pessoas com deficiência nos espaços públicos. Os avanços tecnológicos têm permitido a elas maior autonomia e, consequentemente, uma participação mais ativa no mercado de trabalho e inclusão na vida social. A integração social das pessoas com deficiência representa uma grande conquista: o resgate da cidadania desses indivíduos.

Caberia aqui esclarecer o conceito de *inclusão social*, que é o movimento pelo direito incondicional de que todos os seres humanos participem ativamente da vida pública, sem qualquer restrição de credo, religião, posição política, etnia, opção sexual ou grau de deficiência. Essa definição é diferente da de *integração*, que é o movimento pelo direito de *quase* todos os seres humanos de participarem ativamente da sociedade, desde que estejam devidamente preparados. Embora esses dois vocábulos sejam sinônimos no dicionário, têm conotações diferentes quando se trata de temas ligados a pessoas com deficiência. O termo *inserção social* não está associado a qualquer ideologia e pode, portanto, ser utilizado quando não se deseja falar especificamente de integração ou de inclusão.

As instituições (empresas privadas e órgãos públicos, assim como as universidades), de um modo geral, têm um papel importante na transformação da sociedade e a percepção que as pessoas têm a respeito delas é construída com base nas ações adotadas. Além da motivação legal e ética, uma política de inclusão das pessoas com deficiência, além de outras de caráter social, certamente trazem ganhos significativos de imagem a essas instituições. O clima organizacional também melhora, estimulando o espírito de equipe dos funcionários, gerando sinergia em torno de um objetivo comum e humanizando o ambiente de trabalho, além de possibilitar ganhos de produtividade, se as pessoas com deficiência estiverem devidamente inseridas em funções que otimizem o seu desempenho. Em síntese, a contratação e integração de pessoas com deficiência é tida como uma atitude positiva e a instituição passa a ser vista como um modelo a ser admirado e seguido.

1.2 Motivação para o desenvolvimento desta tese

A primeira motivação foi a de conhecer os fundamentos e aplicações da robótica móvel, o que se mostrou insuficiente para um trabalho mais atual e inovador, sendo, então, necessária a vontade de conhecer a inteligência artificial, seus mecanismos e como ela poderia

ser um suporte em projetos que oferecem certo grau de dificuldade, alguns muito complexos para se resolver pelos métodos convencionais.

Outras motivações derivadas foram: a de aplicar a inteligência artificial ao dispositivo e em algo útil, que servisse de referência para outros trabalhos, que pudesse ser utilizado em alguma área e satisfizesse a necessidade de se projetar algo novo ou introduzir melhorias em algo conhecido. Surgiu, então, a ideia de que este trabalho seria importante para direcionar e aplicar a tecnologia a uma área na qual a desabilidade física necessitasse de um apoio tecnológico maior e existissem limitações nos atuais suportes aos desabilitados físicos. A implementação de uma estratégia de controle de uma cadeira robótica e equilibrista, com seus comandos feitos de forma vocal, é motivadora, principalmente porque dará um retorno por meio da prática de toda a teoria estudada, tornando-se um laboratório real para futuras propostas.

1.3 Contribuições científicas do presente trabalho

Na revisão bibliográfica, foi possível constatar que existe material literário, sob a forma de artigos e teses, disponível que aborda da forma convencional e já bastante explorada a elaboração de um projeto ótimo de sistemas híbridos para o controle aplicado a cadeira de rodas. Apesar de esse assunto ter sido bem explorado, a presente Tese irá preencher uma lacuna científica, pois não se tratou de forma integral a determinação do Centro de Gravidade (CG) de uma cadeira e de seu usuário. Embora este estudo tenha sido utilizado apenas para a simulação por Espaço de Estados, porque a simulação por Lógica “Fuzzy” não necessita desta determinação, chamou a atenção para o fato de que o CG interfere de forma significativa na dinâmica de sistemas e afeta diretamente o modelo matemático desenvolvido, sendo normal, nos trabalhos revisados, seus autores preferindo soluções mecânicas mais caras de movimentação do cadeirante e que comprometem a confiabilidade do projeto.

O trabalho que se apresenta é um texto científico, de caráter aplicado, com benefícios tecnológicos imediatos, e que, combina de uma forma organizada, princípios e técnicas de áreas distintas, como a mecânica, a eletrônica e a inteligência computacional, todas correlacionadas.

Este estudo tem a natureza de apenas referenciar as teorias convencionais do controle, como já foi dito, excessivamente exploradas nas divulgações científicas, tomando uma variante não convencional baseada em algoritmos híbridos de Inteligência Artificial e que, apesar de diversos outros artigos científicos publicados até quando se divulgou o

primeiro artigo deste estudo, no Congresso do IBERDISCAP2000, realizado no ano 2000 (RENNO; BASTOS FILHO, 2000), na cidade de Madrid, Espanha, no CONEM 2012, na cidade de São Luis do Maranhão, e no EDAS2013, na cidade do Rio de Janeiro, foi comprovadamente inédito. Por ter certo pioneirismo no projeto mecânico voltado para o deficiente físico, justifica-se um trabalho de Tese de Doutorado no tocante às adaptações e adequações das ferramentas baseadas em Inteligência Artificial, nas práticas adotadas e na versatilidade que foi dada neste pelo uso de um controle hibrido baseado em Inteligência Artificial.

A proposta original foi mantida, mas, após uma modificação no projeto, um novo protótipo da cadeira de rodas foi construído. Devido às limitações de recursos financeiros para o Departamento de Engenharia Mecânica da Escola de Engenharia da Universidade Federal de Minas Gerais e às mudanças de critérios pela Universidade e pelo atual governo, não houve possibilidade de continuar a construção do protótipo e os trabalhos puderam seguir apenas o caminho da simulação com bastante atraso. Mas a cadeira, fisicamente considerada, está semifuncional, com possibilidades de ser concluída, de forma limitada no momento.

O presente trabalho, voltado para a aplicação da engenharia mecânica baseada neste campo da automação que denominamos popularmente de robótica no apoio ao deficiente físico tetraplégico, além de introduzir modificações no projeto da cadeira em si pelo sensoriamento, pela distribuição e simetria das massas, irá disponibilizar mais uma metodologia de projeto não convencional, ainda não difundida e alternativa aos métodos tradicionais de projeto de controle devido a uma simplificação pela utilização de recursos da Inteligência Artificial dentro de um acoplamento Neuro-Fuzzy apresentado, portanto, uma metodologia de projeto acessível a vários grupos, do Controle Aplicado a dispositivos não lineares, inerentemente instáveis, mas controláveis.

1.4 Proposta deste trabalho

O objetivo deste trabalho não é o de estudar simplesmente dispositivos e integrá-los, como o fez muito bem o senhor Deam Kamen (McCAA, 2013), mas o de criar uma metodologia científica dessa nova configuração sob um controle inteligente, que permita à cadeira de rodas robótica e equilibrista acessar locais que devem ser habituais ao seu usuário tetraplégico no seu dia a dia. Assim, torna-se o desabilitado físico menos dependente, devido ao uso da tecnologia, e se estende a sua capacidade, o que deve ser uma das aplicações mais nobres do estudo de Robôs.

Inclinar em relação ao plano horizontal, através de uma cadeira de rodas, implica em um estudo mais elaborado sob o aspecto dinâmico (pelas mudanças na inércia e perturbações de origens internas e externas), resultando em estudos comparativos deste controle não convencional com aquele por Espaço de Estados e na necessidade adaptativa deste através do CG, centro de gravidade (mutável). E, como foi dito, por meio da inércia, visto que o desenvolvimento matemático se torna mais exigente, principalmente se forem considerados os aspectos não lineares e de instabilidade da cadeira de rodas envolvida (RENNO, C. A.; LIMAII, E. J.; PINTO R. L. F, 2012).

Pretende-se, com este trabalho de Tese, projetar e, dentro das restrições existentes, desenvolver, desde o projeto mecânico até a modelagem e o controle, uma cadeira de rodas robótica e equilibrista baseada no principio do Pendulo Invertido, mas adaptando-o. Dessa forma, o usuário paciente tetraplégico (variável e instável) será incluído como a massa menor ***m*** e a cadeira, como a massa maior ***M***. Isso já cria uma sutil modificação nas equações originais do Pendulo Invertido, devido às mudanças do centro de massa e do momento de inércia.

A simulação sob a forma do controle convencional (clássica/moderna) será usada apenas como referência comparativa, mostrando que este tem um custo mais elevado, conforme se pode ver em maiores detalhes no apêndice 2. Conforme artigo dos professores Umberto Souza da Costa e Natália dos Santos Lucena Neta, publicado sob a forma de artigo em Metrópole Digital, o custo exigido pela maioria dos algoritmos, geralmente, pode ser definido matematicamente em função de seu principal dado de entrada, o qual se denominou *parâmetro primário* do algoritmo. Fizeram-se considerações sobre o tamanho ***N*** do parâmetro primário, sendo que este tamanho corresponde ao número de elementos que compõem o parâmetro primário. Por exemplo, se o parâmetro primário de um algoritmo for um vetor de inteiros, ***N*** representaria o número de elementos desse vetor. Portanto, o tamanho do parâmetro primário afeta diretamente o custo de execução do algoritmo, sendo diretamente proporcional a ele. O objetivo da análise matemática é expressar a necessidade de recursos de um programa (tempo de execução e/ou espaço de memória) em termos desse parâmetro primário ***N*** (SZWARCFITER; MARKENZON, 1994; EDGEWICK, 2004).

O enfoque deste trabalho fica nos estudos de modelagem e do controle não convencional (contemporâneo) baseado em Inteligência Artificial e de custo mais baixo, que se distancia, de certa forma, da metodologia clássica de controle, segundo o qual uma Rede Neural é capaz de aprender comandos vocais e, então, modificar a base de regras de um

controlador "Fuzzy", trabalhando de forma híbrida sob esta outra forma de controle não convencional, gerando o que se chama de um controle do tipo Neuro-Fuzzy.

Com base nisso, foi proposto um novo tema, na área de pessoas com necessidades especiais, que deveria ser o de uma cadeira de rodas robótica e equilibrista, com a sua navegação comandada da forma vocal pelo próprio paciente, de forma preliminar, beneficiando os tetraplégicos sob um controle Neuro-Fuzzy.

1.5 Desenvolvimento do trabalho

Dando sequência a este trabalho, existem apêndices interessantes ao final que trazem complementos dessa ideia e fundamentações teóricas que podem ser úteis a outros trabalhos. No capítulo 2, enumeram-se os objetivos gerais e específicos pretendidos. No capítulo 3, é apresentada uma revisão bibliográfica atualizada em cadeiras de rodas das principais técnicas de controle convencional e de controle não convencional com a utilização de inteligência artificial, além de possuir uma farta fundamentação teórica em seu apêndice. No capítulo 4, apresenta-se a modelagem e soluções obtidas através de simulações utilizando-se o procedimento proposto, por meio do qual se realiza as modelagens matemáticas e o desenvolvimento da cinemática e da cinética e, adicionalmente, para formar a ideia de um controle por Espaço de Estados adaptativo, formando o cálculo do Centro de Gravidade (CG) em tempo real. Ao longo do capítulo, são discutidos os modos e os modelos empíricos para a previsão de distúrbios e é realizado um estudo matemático preliminar da cadeira em questão, com enfoque na determinação de um CG_{TOTAL} combinando o da cadeira com o do usuário que, conforme se vê no desenvolvimento, afeta o controle convencional, mas não o não convencional. O capítulo 5 apresenta a integração de vários algoritmos que leva a uma estratégia de controle que permite ao sistema operar de forma estável, eliminando ruídos e distúrbios, ao se adaptar e agir.

Além disso, como perspectiva, apresenta-se a solução do problema através de rotinas computacionais, implementáveis para a aplicação da estratégia de controle, que constam dos apêndices e anexos. Realiza-se uma abordagem comparativa do controle convencional, levando-se em consideração a metodologia de Espaço de Estados, assim como do controle não convencional, comparando-se com várias metodologias em que o controle sob a forma moderna convencional e o controle sob a forma não convencional são apresentados, assim como os fundamentos teóricos e práticos necessários para o dimensionamento e projetos baseados em Redes Neurais e na Lógica "Fuzzy".

Na sequência, apresentam-se as características construtivas específicas de cada dispositivo de controle. No capítulo 6, são também apresentados resultados experimentais obtidos para o controle de uma planta simulada, mas já parcialmente construída em laboratório. Propõe-se uma metodologia para o projeto de sistemas ótimos híbridos com a utilização das técnicas de controle abordadas anteriormente, quando se definiu as estratégias de controle avançadas e a opção de realizar tal controle sob a forma não convencional, baseado em Inteligência Artificial híbrida devido a sua versatilidade e ao baixo custo de processamento comparado ao convencional.

Para cada técnica específica, a metodologia propõe uma utilização organizada das equações teóricas e empíricas pertinentes. O capítulo 7 apresenta as considerações finais relevantes e as principais conclusões deste trabalho. Além de algumas sugestões, apresentam-se os resultados do desenvolvimento das estruturas da cadeira na prática e, finalmente, as conclusões, nas quais se encontram propostas deste desenvolvimento para trabalhos futuros.

2 OBJETIVOS

O fato dessas cadeiras de rodas robóticas se restringirem à navegação no plano horizontal e aos paraplégicos limitava em muito a segurança e o acesso de outros tipos de desabilitados físicos mais graves a lugares que normalmente teriam que acessar ou manobrar no seu dia a dia.

O tema objeto deste estudo já estava muito explorado sob o aspecto de navegação no plano sob o controle convencional, incluindo-se, aqui, sensoriamentos diversos.

A metodologia proposta para o desenvolvimento deste trabalho visou simplificar a metodologia baseada em critérios científicos excessivos, levando à opção de se desenvolver um produto aparentemente já consolidado, mas inovador, no conjunto de funções que agrega. Para isso, utilizaram-se ferramentas matemáticas conhecidas, embora adaptadas para a um caso particular, entre elas, a inteligência artificial, partindo-se de um conceito simples (de Engenharia) e evoluindo-se, ultrapassando-se, no que ainda fosse possível, o estado da arte, ou seja, as cadeiras de rodas para desabilitados físicos limitadas ao plano horizontal.

2.1 Específicos

Para a proposição de um procedimento de projeto híbrido de controle através da inteligência artificial, os seguintes objetivos específicos foram realizados:

a) pesquisa do estado da arte das técnicas não convencionais de controle, visando escolher aquele modelo teórico e empírico a ser adotado;
b) cálculo e a determinação do centro de gravidade, CG, em tempo real no espaço tridimensional, de forma não invasiva e natural em função da movimentação do usuário sobre uma placa de sensores baseados em extensiômetros;
c) implementação computacional de um modelo de controle, baseado em inteligência artificial ou Lógica “Fuzzy”, para identificar e possibilitar controlar, em tempo real, um novo modelo experimental de cadeira de rodas;

d) implementação de uma Rede Neural Artificial que possa ser treinada e adaptada para o reconhecimento dos comandos vocais básicos na forma de um algoritmo;
e) aplicação do algoritmo Neuro-Fuzzy para a solução do problema real de navegação do usuário tetraplégico sob a forma de simulação;
f) aplicação da técnica de controle não convencional proposta no procedimento Neuro-Fuzzy desta planta experimental, incluindo-se, aqui, como contribuição tecnológica complementar, a solução construtiva que minimiza os efeitos da inércia pela utilização de otimizações em seu projeto mecânico e no cálculo do posicionamento do CG.

3 REVISÃO BIBLIOGRÁFICA

Alguns dos artigos analisados préviamente (2000-2003) para a elaboração deste estudo tomaram como referência a cadeira de rodas robótica em um plano que, embora citados posteriormente, não é o objeto desta pesquisa. Vários outros fizeram a abordagem fora do plano, o que também foi realizado neste trabalho. Mas foram localizados poucos estudos que trabalharam em um modelo no sentido de integrar estes movimentos como foi feito nesta tese.

Em relação à inteligência artificial, muitos trabalhos interessantes são aqui citados e constam das referências ao final na sequencia desta tese, mas nenhum exatamente igual à estratégia Neuro-Fuzzy, abordada na época deste trabalho. Já em artigos independentes encontram-se conexões que levam a certa similaridade com o projeto que se apresenta, mas nada igual ao que motivou a continuidade desta pesquisa, pois o caminho até a conclusão, embora simplificado por se basear em uma proposta metodológica, fortuitamente foi inédito.

3.1 Cadeira de rodas servoacionada ou robótica

O conceito de aprendizado supervisionado em “Perceptrons” multicamada, com base na técnica do gradiente descendente. Alguns problemas e inconvenientes do procedimento original de aprendizagem “Backpropagation” são discutidos, o que levou ao desenvolvimento deste artigo de tecnicas mais sofisticadas, que se concentra em estratégias de aprendizagem adaptativa. Alguns dos algoritmos de aprendizagem mais populares são descritos e discutidos de acordo com a sua classificação em termos de estrategias de adaptação. O comportamento global e local de vários procedimentos de aprendizagem sobre alguns problemas da convergência de programas “benchmark” populares é relatada, indicando, assim, a escala de robustez e propriedades dos respectivos algoritmos (RIEDMILLER, 1994). Neste estudo, foi feita uma observação ao custo de processamento.

Uma cadeira de rodas para pessoas com deficiência física desenvolvida com um sistema de reconhecimento de voz dependente do usuário, de ultrasom e sistemas de sensores infravermelhos, integrados (MAZO *et al.*, 1995).

Várias abordagens para a aplicação da Lógica “Fuzzy” para robôs móveis autônomos são consideradas. Além disso, foi citado o interesse que muitos pesquisadores

apontavam para a Lógica "Fuzzy" no controle de robôs móveis (OLLERO; ULIVI; CUESTA, 2003).

Estudos sobre a aplicação de tecnologias assistivas e técnicas de controle em cadeiras de rodas (BECKER, 2000).

Estudo em curso sobre uma cadeira de rodas interativa inteligente. Também foi descrito um novo conceito de informação multimodal para o controle de navegação da cadeira de rodas usando uma rede neuro-fuzzy (XUEEN; TIENIU; XIAOJIAN, 2000).

Este trabalho apresentou o suporte teórico e os resultados experimentais de aplicação de técnicas avançadas e inteligentes de controle para o controle rígido e sistemas de rastreamento de trajetória em uma cadeira de rodas robótica (ESPINOSA *et al.*, 2001).

Este trabalho descreveu os resultados em um sistema reativo do tipo "Shared-Control", que permitia uma navegação semiautônoma de uma cadeira de rodas em ambientes desconhecidos e dinâmicos (PIRES; NUNES, 2002).

Resultados em modulação de pulso (eletrônica); circuito de comutação; cadeiras de rodas; baterias e circuitos eletrônicos (HAMANAKA, 2002).

Cadeiras de rodas; servomecanismos; sistemas de controle por realimentação; bioengenharia, são geralmente a única solução disponível que visa resolver problemas de mobilidade. Contudo, até agora não foram eficientes em trazer para pessoas em condição de deficiência motora a completa integração à sociedade. O trabalho visou minimizar problemas de lesão por esforços repetitivos ocorridos na tentativa de transpor barreiras arquitetônicas, como rampas e terrenos irregulares, em especial para as crianças (LOMBARDI, 2002).

Análise de valor (controle de custo); ergonomia; desdobramento da função qualidade; cadeiras de rodas; deficientes fisicos e estabilidade. Em tal estudo, a cadeira de rodas é provavelmente a mais importante ferramenta na reabilitação social de pessoas com deficiência física. A cadeira de rodas convencional é composta de quatro rodas de larguras estreitas. Duas delas, com diâmetro grande, estão montadas no eixo posicionado abaixo da porção do assento da cadeira. As outras duas, com diâmetro menor (normalmente denominado Castors), podem ser posicionadas em frente ou atrás daquelas de diâmetro grande. O objetivo desta pesquisa é analisar as possibilidades de se motorizar cadeiras de rodas convencionais, visto que as cadeiras de rodas manuais representam a maioria (~75%) das cadeiras de rodas existentes (ALVARENGA, 2002).

Este trabalho apresentou um algoritmo de navegação para uma cadeira de rodas em ambiente doméstico semiestruturada. O usuário definia a posição final e a orientação da

cadeira de rodas e o controlador de forma autônoma executava (STEFANOV; AVTANSKI; BIEN, 2004).

Descrito um sistema de controle de Lógica “Fuzzy” para uma cadeira de rodas inteligente voltado para a assistência pela deficiência severa, mas baseado em uma simulação de computador de navegação para cadeiras de rodas, em que a Lógica “Fuzzy” permitia o controle do processo (ŠPACAPAN; KOCIJAN; BAJD, 2004).

Um trabalho se caracterizou como descritivo, pois contemplou o delineamento das condições individuais das crianças/adolescentes investigadas e do ambiente onde habitavam; e da avaliação das condições dos equipamentos de mobilidade sentada, carrinho ou cadeira de rodas, por eles utilizados. Para sua realização, foram selecionados 33 indivíduos que, nos últimos 3 anos, receberam uma cadeira de rodas adaptada, prescrita por profissionais da saúde, especialistas em tecnologia assistiva (GALVÃO, 2006).

Aplicações na área da robótica móvel nas quais, além da navegação autônoma do robô, é necessário que um usuário humano interaja no controle de navegação do robô. Nesse caso, considerado como controle semiautônomo, o usuário humano tem a possibilidade de alterar localmente a trajetória autônoma previamente planejada para o robô. Entretanto, o sistema de controle inteligente do robô, por meio de um módulo independente do usuário, continuamente evita colisões, mesmo que para isso os comandos do usuário precisem ser modificados. Essa abordagem cria um ambiente seguro para navegação que pode ser usado em cadeiras de rodas robotizadas e veículos robóticos tripulados nos quais a segurança do ser humano deve ser garantida. Um sistema de controle que possua tais características deve ser baseado numa arquitetura adequada para robôs móveis, que integre a entrada de comandos de um ser humano com a camada de controle autônomo do sistema que evita colisões com obstáculos estáticos e dinâmicos e conduza o robô em direção ao seu objetivo de navegação (GRASSI, 2006).

Arquitetura de controle híbrida (deliberativa/reativa) para um robô móvel com interação humana. Essa arquitetura, desenvolvida principalmente para tarefas de navegação, permite que o robô seja operado em diferentes níveis de autonomia, possibilitando que um usuário humano compartilhe o controle do robô de forma segura enquanto o sistema de controle evita colisões. Além disso, o plano de movimento do robô é representado por uma função de navegação, sendo proposto um método para combinar um comportamento deliberativo que executa o plano de movimento, com comportamentos reativos definidos no contexto de navegação e entradas contínuas de controle provenientes do usuário. O sistema de controle inteligente definido por meio da arquitetura foi implementado em uma cadeira de

rodas robotizada. Foram apresentados alguns dos resultados obtidos por meio de experimentos realizados com o sistema de controle implementado operando em diferentes modos de autonomia (GRASSI; OKAMOTO, 2006).

Tal trabalho preocupou-se com o levantamento e a estabilização de uma cadeira de rodas, no qual o torque é aplicado nos motores da parte posterior da cadeira de rodas, para forçar uma energia suficiente para levantar esta cadeira para a posição vertical (GHAROONI; AWADA; TOKHI, 2006).

Um estudo de estabilidade e robustez de controladores nebulosos. Um simulador foi desenvolvido para controlar um robô, deslocando-se por um mundo virtual, que contém obstáculos randomicamente distribuídos, sendo que o modelo foi implementado de maneira exitosa. Regras foram removidas deste sistema para simular situações nas quais o projetista não possui todo o conhecimento sobre um determinado problema ou em momento de falha dos controladores. Os experimentos realizados indicam que os sistemas de controle baseados em Lógica Nebulosa podem funcionar adequadamente, mesmo sob condições adversas (MORATORI, 2006).

Nesse trabalho, foi desenvolvido um laboratório virtual para dinâmica veicular e modelagem do contato roda-piso que foi usado para a visualização do comportamento dinâmico de uma cadeira de rodas em diversas situações. Utilizando o ambiente Working Model 3D® para a integração das equações de movimento e visualização dos deslocamentos e movimentos associados, implementou-se modelos do contato roda-piso a partir da literatura de referência (SILVA, 2007).

Uma arquitetura baseada em comportamentos hierárquicos e difusos para sistemas autônomos de guiagem e navegação, em veículos agrícolas autônomos (VAA's) e de robôs agrícolas móveis (RAM's). Entretanto, um número limitado de trabalhos tem proposto sistemas robustos baseados em arquiteturas robóticas capazes de realizar operações múltiplas e independentes, bem como se adaptar às mudanças ambientais no campo. Por outro lado, em outras áreas de pesquisa, um número representativo de arquiteturas baseadas em comportamentos tem sido proposto para guiagem e para navegação autônomas de robôs móveis em ambientes não estruturados e/ou não explorados (SOUSA, 2007).

Neste trabalho, uma arquitetura robótica baseada em comportamentos é desenvolvida para guiagem e navegação de RAM's e VAA's. Regras difusas são utilizadas para compor e coordenar comportamentos primitivos e complexos. O desenvolvimento incluiu: a implementação e a simulação da arquitetura em um minirobô; a avaliação e a caracterização de sensores para o módulo perceptivo da arquitetura; e a aplicação de um

método de análise baseado em um modelo matemático para auxiliar a composição de uma rede de comunicação digital baseada no protocolo CAN para sistemas de controle robóticos. Experimentos foram realizados para avaliar os comportamentos implementados e para avaliar a capacidade de operação da plataforma robótica em um ambiente agrícola simulado. Os resultados mostraram a viabilidade da abordagem proposta. A modularidade da arquitetura com a utilização de controladores difusos descentralizados simplificou a implementação da arquitetura. (SOUSA, 2007).

Estudo que possui foco no desenvolvimento de interfaces e sistemas de navegação para cadeiras de rodas, provendo soluções para a inclusão desse grupo de usuários. Para tanto, a revisão bibliográfica envolve três aspectos: as interfaces para usuários, os sistemas de navegação e a integração entre ambos. Primeiramente, localiza-se a interface como parte da relação entre usuário e tarefa, para, posteriormente, restringir-se a interfaces para portadores de severas limitações físicas. Reconhecendo que determinados tipos de limitações de usuários demandam sistemas de navegação, estes são descritos na forma de exemplos de implementações (MADEIRA, 2008).

As cadeiras de rodas manuais convencionais exigem esforço muscular do usuário nos aros de propulsão, sendo muitas vezes um mecanismo ineficiente na superação das barreiras arquitetônicas. Tal mecanismo alternativo é dotado de uma combinação de alavancas propulsoras e embreagem seletiva que permite o travamento das rodas traseiras da cadeira de rodas no avanço de superação de rampas e rodas livres no momento da propulsão convencional. Foi utilizado o ambiente de Working Model 3D® para a simulação da cadeira de rodas no momento da progressão da superação de rampas. A partir de uma revisão bibliográfica dos padrões de propulsão, desempenho e de mecanismos alternativos existentes, elaborou-se a proposta do mecanismo alternativo (SILVA, 2009).

Análise Crítica dos Produtos de Mobilidade Sentada- Cadeiras de Rodas Utilizadas por Crianças é considerado, na reabilitação social, um instrumento de grande relevância para as pessoas com deficiência física. Tal estudo visa posicionar a questão e desenvolver observações sobre técnicas de readaptação de determinados modelos de acordo com a necessidade do usuário. Descreve características dos quatro modelos pesquisados através de busca em catálogos, com intuito de conhecer suas vantagens e desvantagens de uso. Foram apresentados alguns aspectos ergonômicos que devem ser considerados em projetos de cadeiras de rodas (AMORIM, 2009).

Um estudo teórico exploratório sobre o projeto sistemático de sistemas de assentos especiais para cadeiras de rodas, com ênfase em seus aspectos conceituais e

funcionais. De acordo com os princípios da tecnologia assistiva, que consiste no uso de tecnologias para o atendimento das necessidades de pessoas com deficiência, o objetivo das ajudas técnicas deve ser a promoção da atividade e participação social através da compensação de deficiências, promovendo-se a funcionalidade. Assim, fez-se uma revisão da literatura a respeito da adequação postural do usuário de cadeira de rodas e dos fatores que contribuem para sua saúde, sua funcionalidade e seu conforto ou desconforto na posição sentada, permitindo a identificação de suas necessidades explícitas e implícitas (MORAES, 2009).

Um produto orientado para facilitar a mobilidade de sujeitos com deficiências motoras. Segundo o autor, este é um produto feito em Portugal e mundialmente inovador, criado na Universidade do Minho e desenvolvido na SAR. A cadeira seria comercializada pela OrtoMaia, que já venceu três prêmios, o Prêmio Nacional de Inovação, BES Inovação, o prêmio NortiNov, da Agência de Inovação, e o Inventuminho. O projeto teve o seu início em 2007, no seio do Grupo de Automação e Robótica da Universidade do Minho, liderado por Fernando Ribeiro, que esteve em contato com uma equipe de Robótica (RIBEIRO, 2010).

A LabVIEW™, através do seu módulo de *Intelligent Control Systems*, publicou um artigo sobre *Neuro-fuzzy Controller Theory and Application*, no qual sistemas “fuzzy” permitem transferir a forma difusa vaga do raciocínio humano aos sistemas matemáticos. O uso de regras “if-then” em sistemas “fuzzy” permite compreender facilmente o modelo de informação (CRUZ; FIGUEROA, 2010).

Descrevem a implementação de três controladores diferentes para uma cadeira de rodas, que permitem ao usuário total controle sobre esta (PONCE; MOLIN; MENDOZA, 2010).

Xiao e outros autores publicaram em *Advances in Neural Network Research* (2010) um artigo intitulado Visual Navigation of a Novel Economical Embedded Multi-mode Intelligent Control System for Powered Wheelchair, a fim de ajudar os diferentes tipos de pessoas com deficiência e os idosos, além de melhorar a eficácia e conveniência de interação máquina-ambiente humano, incorporando um controle multimodo econômico (XIAO, J. *et al.*, 2010).

Um artigo intitulado *A Modular Fuzzy Control Approach for Two-Wheeled Wheelchair*, no qual citam que cadeiras de rodas em duas rodas estão se tornando parte essencial de pessoas desabilitadas. Mas a concepção de estratégias de controle para tais cadeiras de rodas é uma tarefa desafiadora, devido ao fato de que elas são sistemas altamente não lineares e instáveis. O *design* sutil do sistema imita um pêndulo invertido duplo com três

atuadores, um para cada roda e um para a posição da cadeira e ao universo de estratégias de controle inadequados para a cadeira de rodas de duas rodas. No trabalho desses pesquisadores, o sistema começa a trabalhar com o levantamento das rodas dianteiras (rodízios ou “castors”) para a posição vertical e ainda com a estabilização na posição vertical. Reconheceram que o grande desafio residiu na concepção e implementação de estratégias de controle adequadas para a cadeira de rodas de duas rodas, a fim de realizar de forma comparativamente semelhante a uma cadeira de rodas normal de quatro rodas. Um controlador modular lógica “fuzzy” de dois níveis é proposto neste trabalho. Um modelo de cadeira de rodas padrão também é desenvolvido como um teste e uma plataforma de verificação usando o software Visual Nastran integrado com Matlab (AHMAD; SIDDIQUE, 2011).

Sistema de reconhecimento de voz que se aplica ao controle do aparelho eletromecânico de voz, especialmente robôs móveis controlados por voz ou cadeira de rodas inteligente para as pessoas portadoras de deficiência. O objetivo era interagir com o robô usando técnicas de comunicação naturais e diretas. O escopo deste trabalho foi a forma como a voz pode ser processada para obter o movimento adequado e seguro da cadeira de rodas com uma elevada taxa de reconhecimento. A fim de tornar a voz uma ferramenta de comunicação eficiente entre humanos e robôs, alta taxa de reconhecimento de voz deve ser alcançada. Mas a taxa de reconhecimento de voz de cem por cento, em um ambiente em geral, é muito difícil de ser obtida. No trabalho citado, a técnica proposta, chamada transformação “Multiregdilet”, é utilizada para o reconhecimento de palavras isoladas. Finalmente, usa-se as saídas de rede neural (NNT) para controlar a cadeira de rodas através de “notebooks” (computador) e interface especial de “hardware”. A taxa de sucesso do reconhecimento de 98% foi alcançada (AL-THAHAB, 2011).

Sistema de assistência multimodal controlado por voz com controle difuso. Os comandos de voz são muitas vezes a maneira mais conveniente para controlar diversas ferramentas de apoio e para obter a funcionalidade completa de comandos de voz (RUDZIONIS; MASKELIUNAS; RASYMAS, 2012).

Um artigo intitulado Implementation of Fuzzy Logic Controller in FPGA Circuit for Guiding Electric Wheelchair e este trabalho descreve uma implementação do sistema de controle de Lógica “Fuzzy” para orientar uma cadeira de rodas. Para esse fim, uma arquitetura dedicada de controlador de Lógica “Fuzzy” foi elaborada no circuito FPGA (POPLAWSKI; BIALKO, 2012).

Trabalho que propõe um novo conceito de cadeira de rodas manual ergonômica com assistência motorizada servo-controlada. A mobilidade em cadeira propulsionada manualmente é uma das formas mais utilizadas de locomoção entre pessoas com deficiências, porém expõe o usuário ao risco de lesão nos membros superiores e não propicia independência em todas as situações de locomoção diária. A primeira alternativa tecnológica aplicada pelo homem foi a motorização. No entanto, o uso dela não implica em melhora da mobilidade, pois leva o usuário ao sedentarismo. Nesse sentido, fazendo a integração do histórico, ergonomia, biomecânica e engenharia mecatrônica, este projeto objetivou o desenvolvimento do projeto conceitual ao protótipo de uma cadeira de rodas manual com assistência motorizada complementar e proporcional à força propulsora aplicada pelo usuário (MEDOLA, 2013).

O delineamento desse projeto de Fausto Orsi Medola e Valéria Meirelles Carril Elui envolveu o levantamento das evidências científicas relacionadas aos aspectos da configuração de cadeiras manuais determinantes para a melhoria do desempenho na mobilidade e conforto. A criação do conceito abrangeu, ainda, a realização de estudos complementares dos aspectos mecânicos, ergonômicos e cinemáticos da locomoção em cadeiras manuais. O conceito do protótipo tem seus pilares na: propulsão, através de um servo-motor centralizado que atua nas duas rodas traseiras, por meio de um diferencial mecânico, simplificando o controle da assistência motorizada; no projeto do aro propulsor ortogonal com acoplamento assimétrico e a roda em cambagem, proporcionando melhor acesso das mãos ao aro; no projeto do aro propulsor, baseado em aspectos dactilo-anatômicos; no projeto e configuração da cadeira, estruturada nas evoluções tecnológicas das cadeiras manuais e adaptada às características antropométricas da população brasileira; na assistência motorizada controlada por um sistema de avaliação contínua da inclinação do terreno; nas forças aplicadas ao aro e o deslocamento angular das rodas medido por acelerômetro, células de carga e encoder ótico, respectivamente. A efetivação do presente conceito favorece a mobilidade independente do cadeirante, podendo reduzir o custo referente ao uso prolongado de cadeiras manuais: lesão dos membros superiores e incapacidade funcional. Aprimorar a mobilidade de usuários de cadeira de rodas favorece sua independência e participação social, aspectos essenciais para a melhora da qualidade de vida (MEDOLA, 2013).

As contribuições deste estudo foram a escolha de um modelo matemático para a CRM com incerteza quanto à posição do Centro de Gravidade (CG) dela; o desacoplamento dos sistemas dinâmico e cinemático; a linearização dos modelos; a construção das matrizes de incertezas (vértices do politopo); a proposta da malha de controle; a obtenção das matrizes

que tornam o sistema ERP e que estabilizam o sistema com uma taxa de decaimento especificada; as matrizes que tornam o sistema ERP e que estabilizam o sistema a partir da metodologia de Controle com Estrutura Variável (CEV) via LMIs, sendo desenvolvida a demonstração para sistema com número de saídas maior que o número de entradas, com incertezas paramétricas e com taxa de decaimento especificada (ROSSINI, 2013).

3.2 Controles com técnicas híbridas

Pode-se dizer, de acordo com a revisão bibliográfica realizada, que existe a possibilidade de associações entre os diversos tipos de controle no que tange a uma melhor estratégia. Para o controle que se denomina convencional, seja ele clássico, moderno, ou outro, inúmeros trabalhos realizados provaram sua eficácia quando se uniram os vários tipos para se formar um controle híbrido. Os resultados demonstram que esse controle híbrido consegue superar aquele tipo único. Sob esse aspecto, foi interessante descobrir que a lógica nebulosa e redes neurais, por exemplo, são tecnologias complementares no projeto de sistemas inteligentes voltados para o controle que associadas, possuem mais méritos que deméritos. Outros algoritmos como genéticos e imunizantes, existem, mas são apenas citados. A TAB. 1, na qual foram atribuidos pesos convertidos e normalizados de forma numérica (1-10), mas baseados nos diversos conceitos mencionados da bibliografia de artigos que citavam classificações tais como: “Muito bom”, “bom”, “ruim”, demonstra que essas associações se tornam mais robustas em termos de peso. Isso foi colocado na TAB. 1, a seguir, embora proporcione uma ideia relativa, seria necessário um universo maior de pesquisa para se obter um valor melhor determinado.

TABELA 1 - Pesos comparativos de controles simples e combinados

Campos Tipos	Lógica Nebulosa	Rede Neurais Artificiais	Controle Clássico
Modelo Matemático	4	8	8
Habilidade para Aprender	0	8	0
Representação Conhecimento	8	0	2
Conhecimento Expecialista	8	0	2
Não Linearidade	8	8	0
Habilidade em Otimizar	0	4	2
Tolerância a Falhas	8	8	0
Tolerância a Incertezas	8	8	0
Operação em Tempo Real	8	4	8
Total Individ.	**52**	**48**	**22**
Tot.Int.Parc.1	**100**		
Tot.Int.Parc.2		**70**	

FONTE: Dados da pesquisa.

Para alguns sistemas, como por exemplo, os não lineares, sujeitos a ruídos, a aquisição de regras para a formulação de uma estratégia de controle bem definida certamente não é uma tarefa fácil. Para esses casos, controles nebulosos dotados de capacidade de aprendizado podem ser empregados.

A capacidade de aprendizado pode ser obtida pela associação dos sistemas nebulosos às redes neurais que interferem na base de regras conforme será apresentado adiante. A integração desses dois sistemas agrega as vantagens de ambos (ver TAB. 1), isto é, as redes neurais dotarão o sistema resultante com a capacidade de aprendizado e os sistemas nebulosos dotarão o sistema hibrido resultante com a capacidade de decisão.

3.3 Uma comparação entre a Segway e a cadeira de rodas

Outras alternativas para o transporte em pequenas distancias baseiam-se no controle de um veículo fundamentado no pêndulo invertido, no qual o usuário fica de pé (ou sentado, como na cadeira de rodas) sobre duas rodas paralelas, mantendo-se em equilíbrio instável sob a ação dos motores. Embora, em princípio, seja uma opção um tanto forçada, essa classe de dispositivo é única, ocupa quase o mesmo espaço que uma pessoa de pé, bem como ter uma curva de aprendizagem operação muito rápida, assim uma vez que tudo o que é feito é o equilíbrio como uma pessoa faz ao caminhar. Esses veículos não requerem um grande conhecimento prévio, apenas o suficiente para o transporte. Além disso, uma das grandes vantagens sobre veículos tradicionais é a sua elevada mobilidade, uma vez que é possível obter um raio de viragem de zero, pois ambas as rodas giram em direções opostas e podem facilmente ser mobilizadas por vias complexas, o que só pode ser feito em pé.

Esse tipo de veículo basicamente tem um único expoente, chamado Segway, que foi o primeiro veículo comercial da categoria. Até o momento, há também um casal de trabalho semelhante em conjunto, projetado para controlar um veículo instável com base no primeiro projeto. A seguir, brevemente descrevem-se esses projetos e recursos. Em 2001, o desenvolvedor americano Dean Kamen (HARMON, 2001**)** introduziu este dispositivo revolucionário de transporte para a época. Antes disso, havia projetado a iBot, a cadeira equilibrista já citada anteriormente, mas a Segway é um novo projeto, cujo funcionamento baseia-se no equilíbrio natural das pessoas: o Segway PT[1].

---

[1] *Segway Personal Transporter.*



FIGURA 1- Segway PT

O surgimento desse novo meio de transporte baseado em uma premissa tão estudada, como é o controle de um pêndulo invertido, gerou grande interesse em todo o mundo, pois se tratava de uma solução simples, em termos construtivos, para usar tração elétrica, podendo, eventualmente, contribuir fortemente para descongestionar as ruas das grandes cidades, já que tem o potencial para absorver a demanda de transporte em distancias curtas.

A Segway Inc. oferece diferentes versões do dispositivo, sendo que existiam duas principais linhas de modelos: a linha de base, chamada i2, e a linha off-road, chamada x2 (FIG. 1). A TAB. 2 mostra as principais características do veículo para ambos os modelos:

TABELA 2 - Principais características

Modelo	**Segway i2**	**Segway x2**	**Cadeira Protótipo**
Peso	47.7 [kg]	54.4 [kg]	12,81 [kg]
Tamanho das Rodas	48,3 [cm]	83,8 [cm]	37[cm]
Tamanho da Base	48x63 [cm]	53x84 [cm]	20x40[cm]
Velocidade Máxima	20 [km/hr]	20 [km/hr]	12[Km/hr]
Autonomia	38 [km]	19 [km]	20[Km/hr]
Motores DC	Brushless	Brushless	Normal

FONTE: Dados da pesquisa.

No entanto, o veículo não conseguiu o sucesso comercial esperado, principalmente devido ao elevado custo de acesso a uma unidade (cerca de $5,000 EUA). Assim, esses veículos passaram a aplicações específicas, como a de ser uma substituição aos *caddies* populares em campos de golfe ou aplicações de negócios, como marketing, ou de guarda de segurança em movimento. Ele também conseguiu penetrar em um grupo de elite de usuários cuja renda permitia o acesso a essas tecnologias e um certo nível de compromisso /

fanatismo com tal veículo, como no estilo gerado por produtos eletrônicos em seus consumidores.

A cadeira de rodas possui dois modos de operação:

- Giros e deslocamentos lineares no plano;
- Inclinação(com possibilidade de compor com o modo 1).

Segundo a sua patente, o Segway possui dois modos de operação:

- Estabilização sem condutor (“riderless”);
- Modo movimentação.

Inclinação e estabilização sem condutor:

- Ambos se comportam exatamente como um pêndulo invertido na procura do equilíbrio, mas ao atingí-lo:

1. Na cadeira, o CG é equilibrado e a velocidade pode variar;
2. No Segway, o CG é equilibrado e a velocidade é zero.

- Esta diferença surge na referência ou “input” que na cadeira é variável e no Segway é fixo para sempre equilibrar;
- No Segway, o usuário é a perturbação necessária ao movimento, mas na cadeira não. Ele é efetivamente uma perturbação.

No primeiro semestre de 2009, a Segway Inc. apresentou um projeto experimental, juntamente com a General Motors, chamado projeto PUMA[2]. Esse projeto pode ser observado na FIG. 2, cuja posição revela uma solução de transporte urbano real, fornecendo ao projeto original certas características que o tornam mais amigável. Em particular, esse veículo é baseado no *design* clássico, mas pode transportar dois passageiros sentados, assemelhando-se a um carro. O protótipo vem com rodas extras atrás, usadas nas restrições do veículo, mas o movimento ainda é no sentido de se equilibrar dinamicamente

[2] Mobilidade Urbana e Acessibilidade Pessoal.

sobre as duas rodas do meio. Tal veículo também pode ser uma grande oportunidade para as pessoas com deficiência, permitindo um comportamento natural (inclinando-se) e sem o risco de quedas.



FIGURA 2 - A evolução da Segway denominada P.U.M.A. da GM
Fonte: MILLER (2009).

Assim, apesar da aparente falta de sucesso do veículo, trata-se de uma área interessante de desenvolvimento de veículos pessoais que chamou a atenção de muitos desenvolvedores, criando uma série de protótipos que emulam equilíbrio humano com base semelhante às estruturas do pêndulo invertido, como foi o caso deste trabalho da cadeira robótica e equilibrista, estendendo-se a definição do veículo como um modelo específico de uma empresa a uma categoria de transporte para tetraplégicos.

O autor considera que, para o caso desta cadeira, este seria o caminho natural.

4 MODELAGEM MATEMÁTICA

Todos os fenômenos físicos deste universo são passíveis de modelagem matemática. Alguns deles são mais simples de modelar e outros, tão complexos que um modelo matemático aproximado já se torna um grande feito.

A modelagem matemática, dotada de critérios adequados, proporciona a criação de um conjunto de equações denominadas equações da dinâmica do movimento, ou seja, equações matemáticas que descrevem o comportamento dinâmico do fenômeno físico, como é o caso da cadeira de rodas robótica e equilibrista.

O modelo matemático dessa cadeira, incluindo-se aqui as forças não conservativas, devido ao atrito, embora não seja do tipo mais complexo, situa-se em um grau de dificuldade maior.

Inicialmente, foi realizado o desenvolvimento de um modelo sem restrições (holonômico), representando um comportamento de um objeto no espaço tridimensional com todos os seus graus de liberdade, o que já foi relativamente complexo e a decisão de fazê-lo de forma tão generalista foi para se ter uma noção do comportamento dinâmico de um objeto inerentemente não linear e instável e da geração das equações inerentes a ele. A introdução de restrições da cadeira e a sua posterior adaptação para um modelo mais verdadeiro (não–holonômico) (FIGUEIREDO; JOTA, 2003), com suas restrições, conduziu o restante do desenvolvimento deste trabalho.

O termo holonômico é atribuído a Hertz (ARNOL'D; NOVIKOV, 1994) e significa "universal", "integral", "integrável" (literalmente: holo = o todo, conjunto, totalidade; nomia = lei). Portanto, sistemas não holonômicos podem ser interpretados como sistemas não integráveis.

A abordagem matemática desse tipo de problema é realizada através de ferramentas da geometria diferencial. O desenvolvimento sistemático da teoria iniciou-se há mais de 150 anos, baseado em uma série de artigos clássicos sobre a mecânica não holonômica de matemáticos e físicos, tais como: Hertz, Voss, Hölder, Chaplygin, Appel, Rooth, Woronets, Korteweg, Carathéodory, Horac, Voltera, entre outros. Apesar disso, apenas recentemente (KOLMANOVSKY; MCCLAMROCH, 1995) se iniciou o estudo de problemas de controle para tais sistemas.

Definem-se como não holonômicos, sistemas com dimensão finita nos quais algum tipo de restrição, é imposta a um ou mais estados do sistema. Tais limitações podem

ser provocadas pela conservação do momento angular, condições impostas pela impossibilidade de deslocar em uma ou mais direções, como resultado da imposição de restrições durante o projeto do sistema de controle, pelo fato de o sistema não ter atuadores em todas as direções do espaço do problema, e em várias outras situações (MURRAY *et al.*, 1994; WEN, 1996) indica três classes de sistemas em que se encontram restrições não holonômicas: 1) restrição de não deslize. A condição de não deslizamento ou de rolamento puro significa que a velocidade linear no ponto de contato é zero. Essa restrição é não integrável, isto é, não redutível a uma restrição de posição e, portanto, é não holonômica; 2) conservação do momento angular. 3) Sistemas mecânicos subatuados. Sistemas nos quais a dimensão do espaço de configurações excede o espaço das entradas de controle.

Exemplos de sistemas não holonômicos e da geração de trajetórias para os mesmos, em especial, para a análise do problema da queda de uma gata (sistema com corpos rígidos acoplados) são abordados em Fernandes e Gurvits (1994). Esse problema é particularmente interessante, porque se sabe, com base na mecânica clássica, que o momento angular de uma gata em queda é conservado, então, como uma gata em queda pode mudar sua orientação sem violar a conservação do momento angular? A análise desse problema utilizando-se a teoria de aproximação de Ritz leva ao desenvolvimento de um algoritmo numérico para a determinação de soluções quase ótimas (FERNANDES; GURVITS, 1994).

Sistemas não holonômicos formam uma classe com características especiais: apesar de seus movimentos serem limitados, eles podem atingir qualquer configuração no espaço em que estão definidos (quando controláveis e atingíveis); infelizmente, as leis de controle para estabilização de sistemas não holonômicos não são fáceis de serem geradas; há necessidade de emprego de ferramentas matemáticas mais elaboradas para análise e projeto (geometria diferencial e controle não linear ou linear variante no tempo).

A consideração de restrições ao movimento melhora significativamente o controle de sistemas não holonômicos, possibilitando o projeto de controladores multivariáveis exponencialmente estáveis de forma integrada. O desafio para análise e síntese de controladores para sistemas desse tipo tem propiciado o desenvolvimento da teoria de controle não linear. Técnicas de otimização e controle geométrico empregando transformações de coordenadas (transformações lineares e não lineares) e sinais variantes no tempo ou descontínuos são utilizadas para a geração de trajetórias e projeto de controladores em malha aberta. Em malha fechada, destacam-se: controle adaptativo, robusto, inteligência artificial, linearização por realimentação de estados e das saídas empregando sinais contínuos variantes no tempo ou descontínuos, e técnicas de controle híbrido. As principais técnicas de

controle empregam funções de Lyapunov, associadas a um método para gerar as leis de controle: integrador de um passo atrás (backstepping), método direto, método inverso, métodos matemáticos, métodos utilizando a física do processo e outros. Em seguida, há uma breve revisão bibliográfica sobre pesquisas mais significativas na área.

O trabalho que serve como tutorial para iniciantes no controle de sistemas não holonômicos foi desenvolvido por Kolmanovsky e McClamroch (KOLMANOVSKY; MCCLAMROCH, 1995). O autor apresenta, de forma clara e acessível, os estágios de desenvolvimento da teoria, indicando modelos, técnicas de controle em malha aberta e fechada e métodos de planejamento de trajetórias. Avanços no controle por realimentação variante no tempo são apresentados por Morin e Samson (MORIN; SAMSON, 2000). Uma visão geral sobre o problema da estabilização de veículos autônomos, indicando problemas em aberto e tendências, é apresentada em Aguiar e Pascoal (2000), que destacam a necessidade do estudo de incertezas no modelo e o desenvolvimento de controladores robustos, principalmente para veículos marinhos.

Essa opção foi tomada por ser mais desafiadora, realista, principalmente, e porque, junto à metodologia aqui apresentada, pudesse ser adotada futuramente para outros sistemas similares.

Naturalmente, ao se completar o modelo e pelo fato de a cadeira estar sob várias restrições, o que de certa forma dificulta a visualização do modelo final, pois ele se torna muito complexo, o que é típico de um dispositivo não linear e instável, para que se pudesse visualizar seu comportamento sob determinados movimentos, tal modelo sofre uma simplificação em determinadas equações da dinâmica, considerando-se os sete movimentos dos comandos fundamentais da cadeira, reduzindo-se a um dos modelos do movimento desacoplado como, por exemplo, inclinar em relação ao solo. Esse foi o movimento tomado como referência para que se pudesse analisar posteriormente a outra modelagem matemática, qual seja, para movimentação no plano.

O tipo de metodologia utilizada para a formulação das equações de movimento se torna particularmente importante para a modelagem de sistemas no espaço tridimensional e, seja qual for, essa metodologia deve resultar nas mesmas equações da dinâmica para o mesmo dispositivo.

Posteriormente, essas equações da dinâmica servirão, como apresentado no final de cada metodologia de modelagem, para simular o movimento da cadeira, podendo realimentar correções tanto em sua cinemática como modificações estruturais (como foi o caso do estudo da posição do CG diante da assimetria devida à movimentação do usuário).

Para o caso de um dispositivo generalista, consideram-se, inicialmente, todos os deslocamentos lineares possíveis, que poderiam ser interpretados como um movimento prismático em um robô, assim como todas as rotações possíveis entre dois referenciais, um deles inercial e outro local ou móvel, obtendo um modelo de um dispositivo genérico com seis graus de liberdade sem restrições. A FIG. 3, a seguir, caracteriza tais movimentos.

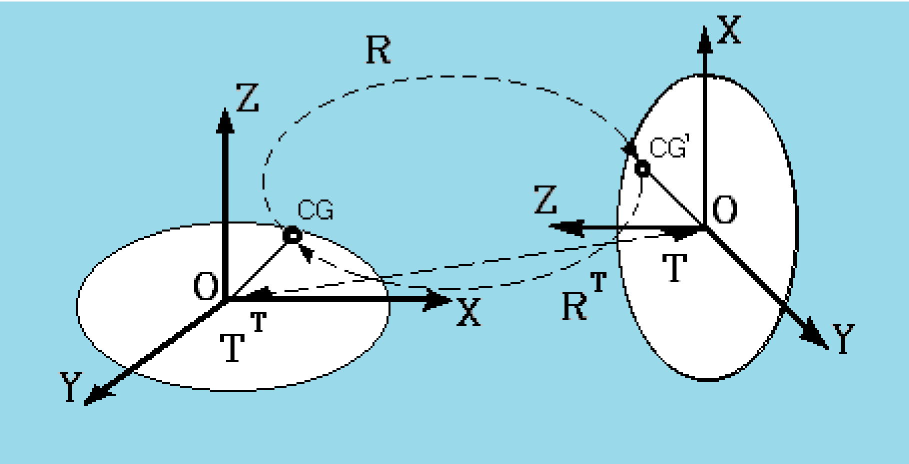


FIGURA 3 - Movimentos do dispositivo generalista
Fonte: Dados da pesquisa.

Supondo um movimento de rotação e de translação sob forma de uma matriz **T** generalizada de um ponto **CG** nas coordenadas x, y e z, em relação ao referencial inercial para um ponto **CG'** de coordenadas x', y' e z' em um referencial local de um corpo rígido no espaço, temos para a cinemática direta:

$$\mathbf{CG' = T\ \ CG} \qquad \text{(eq.4.0.1)}$$

E, para a cinemática inversa:

$$\mathbf{CG = CG'\ T^{-1} = T^{T}\ CG'} \qquad \text{(eq.4.0.2)}$$

Sendo T a nomenclatura utilizada para transformadas generalizadas de translação e rotação do tipo matricial abaixo referenciado:

$$\begin{bmatrix} a_{xx} & a_{xy} & a_{xz} & d_x \\ a_{yx} & a_{yy} & a_{yz} & d_y \\ a_{zx} & a_{zy} & a_{zz} & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Sendo ***a***, transformações referidas a ângulos do tipo θ, β, φ, ... ,(combinados, ou não), e ***d*** deslocamentos (x,y,z). Os valores adicionais dessa matriz estão relacionados às complementações das dimensões, mas são muito utilizados em processos gráficos.

Para o caso da cadeira de rodas em análise, inicialmente, poucas restrições foram feitas. Por isso, não existem os deslocamentos laterais, supondo não haver deslizamentos, e transversais, pois as rodas traseiras estão sempre apoiadas no solo. Um preâmbulo deste estudo pode ser visto através da FIG. 4 a seguir:

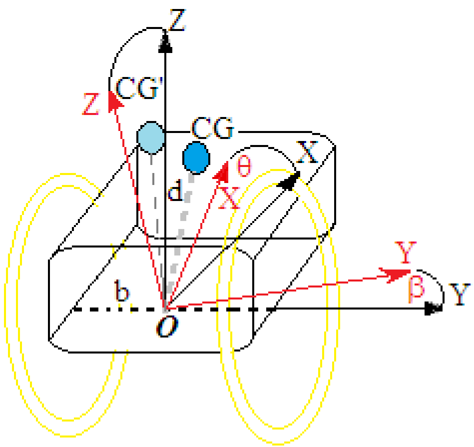


FIGURA 4 - Movimentos da cadeira de rodas no plano e fora dele com restrições
FONTE: Dados da pesquisa.

A representação dos movimentos cinemáticos é uma etapa fundamental para o prosseguimento deste trabalho, embora não seja o objetivo final, como foi dito anteriormente, pois levará à cinética do dispositivo analisado e ao foco principal que é o controle nebuloso.

4.1 O modelo matemático

Inicialmente, segundo a metodologia de Newton/Euler, foi desenvolvido, de acordo com a geometria da cadeira, conforme a FIG. 5.

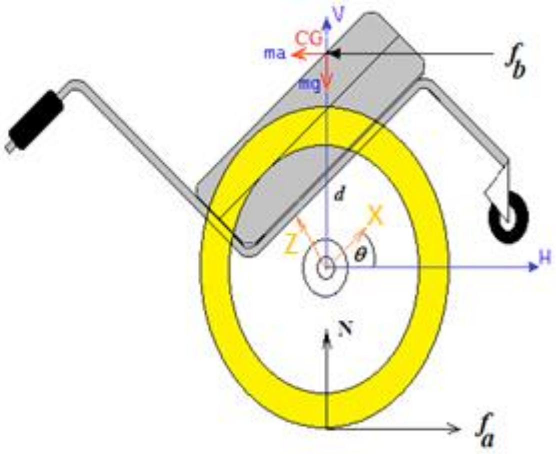


FIGURA 5 - Diagrama geométrico para modelagem da cadeira na inclinação
FONTE: Dados da pesquisa.

4.1.1 Modelo dinâmico da cadeira

Esse modelo se assemelha, como foi dito, ao de um pêndulo invertido, com exceção de alguns pontos, quais sejam: 1) Toda a massa total (m + M) da cadeira foi considerada como estando no CG (m é a massa do usuário e M é a massa do conjunto impulsor, que foram mantidos na equação para mostrar a similaridade com o Pendulo Invertido); 2) A condição inicial para o ângulo Theta (θ) de inclinação do CG calculado em relação ao solo é de 69º e, naturalmente, varia através do movimento do usuário, do seu tipo e do controle para atingir o equilíbrio (OLIVEIRA; SILVEIRA, 2002).

Por meio desse modelo, é possível conhecer o comportamento dinâmico através das suas equações, ou seja, a equação 1 da aceleração linear (a=dv/dt=d2x/dt2) e a equação 2 da aceleração angular (α=dω/dt=d2θ/dt2), em função do Torque T, convertido na ação da força F (denominada de U nas equações 4.1 e 4.2), aplicada às mesmas, pelos motores de forma unificada. As duas equações da dinâmica (no modelo matemático de simulação f(u)) podem ser vistas a seguir:

$$
\begin{aligned}
a = \dot{v} = \ddot{x} = & \\
= & -(-JU + m^2 d^2 \sin\theta \cos\theta g + md \sin\theta \, b_\omega(\dot{\theta}) + f_v(\mp\dot{x})md^2 \\
& - m^2 d^3 \cos\theta \, \dot{\theta}^2 - Jmd \cos\theta \, \dot{\theta}^2 - Umd^2 + Jf_v(\mp\dot{x}) + Jb_v\dot{x} \\
& + md \sin\theta \, f_\omega(\mp\dot{\theta}) + b_v \dot{x} md^2)/m^2 d^2 \cos\theta^2 + md^2 M + JM + Jm
\end{aligned}
$$

(eq.4.1)

$$\alpha = \dot{\omega} = \ddot{\theta} =$$
$$= -\Big(-md\sin\theta\, U + md\sin\theta\, f_v(\mp\dot{x}) + md\sin\theta\, b_v(\dot{x}) - m^2d^2\sin\theta\cos\theta\,\dot{\theta}^2 + f_\omega\big(\mp\dot{\theta}\big)M + f_\omega\big(\mp\dot{\theta}\big)m + d\cos\theta\, mgM + d\cos\theta\, m^2g + b_\omega\big(\dot{\theta}\big)M + b_\omega\big(\dot{\theta}\big)m\Big)/(m^2d^2\cos\theta^2 + md^2M + JM + Jm)$$

(eq.4.2)

Sendo que:

U é a ação unificada de controle (no caso, **F** devido ao Torque **T** na roda de raio **r**);
m é a massa do usuário a ser referida ao CG;
M é a massa da cadeira, mas como **m** referida ao CG;
d é a distância do CG ao seu referencial no tempo t (que muda de acordo c/o usuário);
J é o momento de inércia de uma barra referente ao CG (ver definição na eq.4.6);
θ é o ângulo de inclinação no tempo t;
g é a constante gravitacional e;
$\mathbf{f_x}$ e $\mathbf{b_x}$ são os atritos estáticos e viscosos respectivamente.

Por definição, o momento de inércia ***J*** de uma partícula de massa ***m*** e que gira em torno de um eixo, a uma distância ***r*** dele, é:

$$J = mr^2$$ (eq.4.3)

Este é a referência de inércia adotada em vários trabalhos baseados no Pêndulo invertido.

Se um corpo é constituído de ***n*** massas pontuais (partículas), seu momento de inércia total é igual à soma dos momentos de inércia de cada uma destas massas:

$$J = \sum_{i=1}^{n} m_i r_i^2$$ (eq.4.4)

Sendo $\boldsymbol{m_i}$ a massa de cada partícula, e $\boldsymbol{r_i}$ a sua distância ao eixo de rotação.

Para um corpo rígido, pode-se transformar o somatório em uma integral, integrando, para todo o corpo ***C,*** o produto da massa ***m*** em cada ponto pelo quadrado da distância ***r*** até o eixo de rotação:

$$J = \int_C r^2\, dm$$ (eq.4.5)

Há vários valores conhecidos para o momento de inércia de certos tipos de corpos rígidos (assumindo distribuição uniforme de massa):

Para uma barra delgada, com área de seção transversal tendendo a **0** e comprimento d, perpendicularmente à barra e passando por uma de suas extremidades:

$$J = \frac{1}{3} M d^2 \qquad \text{(eq.4.6)}$$

Essa última relação de inércia foi escolhida para representar as equações da dinâmica da cadeira de rodas robótica e equilibrista por ser a geometria referencial que segundo esta análise, está mais próxima de representar o comportamento do dispositivo.

4.2 Simulação do CG resultante

Para estudar as variações devidas ao tipo e aos movimentos do usuário, para simular o CG humano, o CG da cadeira e a composição dos dois, foi desenvolvido, no Matlab® da MathWorks®, um programa para fazer esta simulação.

De forma semelhante, para se efetuar o cálculo de CG total do conjunto, considera-se a distância de um eixo de referência com a massa de cada parte do corpo e da cadeira e se calculam as coordenadas com as equações fisicas seguintes:

$$x_{total} = \frac{\sum x_i \times W}{\sum W} \qquad y_{total} = \frac{\sum y_i \times W}{\sum W} \qquad z_{total} = \frac{\sum z_i \times W}{\sum W}$$

Sendo que:

$x_{total}, y_{total}, z_{total}$ representam as coordenadas do centro de massa do corpo composto;

x_i, y_i, z_i representam as coordenadas do centro de massa de cada parte que constitui o corpo e a cadeira;

$\sum W$ é a soma dos pesos de todas as partes que constituem o conjunto.

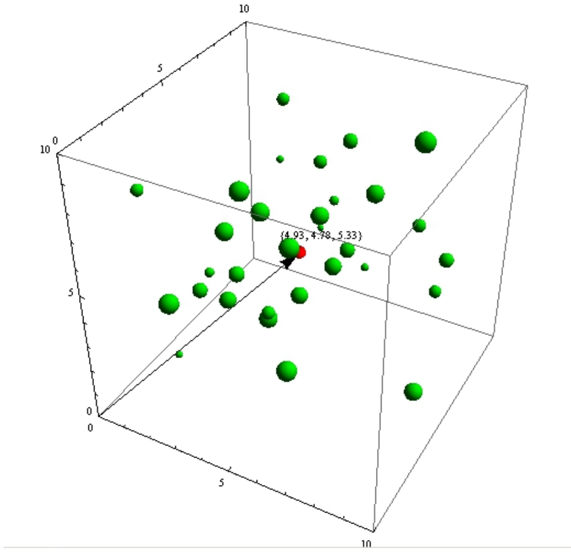


FIGURA 6 - Exemplo de diversos centros de massa e o CG resultante
Fonte: Dados da pesquisa.

Na primeira parte do programa, foi calculado o CG da cadeira, adotando-se o procedimento de usar a massa de cada parte da cadeira multiplicada pela coordenada do CG desta parte divido pela massa total de cadeira. A seguir, definiram-se as coordenadas das diversas massas da cadeira em relação a um eixo de referência. Na sequência, é apresentada a parte da metodologia referente à primeira em que se realizou a Soma do produto e a Soma das massas.

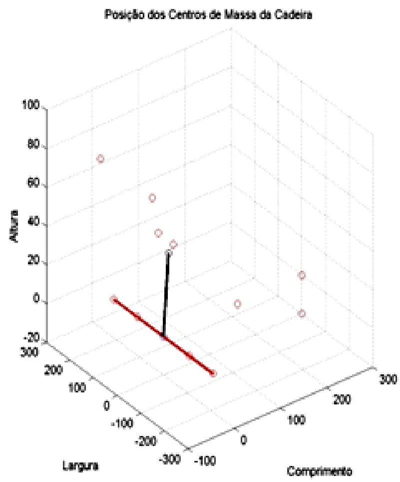


FIGURA 7 - As diversas coordenadas estáticas dos centros de massa da cadeira e o CG resultante
FONTE: Dados da pesquisa.

É efetuado então o produto das massas pelas coordenadas e o resultado é dividido pela massa total. Assim, determinam-se as coordenadas do CG da cadeira.

A segunda parte do programa simula aleatoriamente as variações das forças que atuam nos quatros extensômetros, $S_{1,}$ $S_{2,}$ $S_{3,}$ $S_{4,}$ para simular a movimentação do cadeirante sentado. Em seguida, é efetuado o cálculo de CG humano.

Sequencialmente, identificamos as coordenadas de cada sensor em relação a um eixo de referência.

Nesse trecho do programa, é simulada a variação das forças aleatoriamente e, na sequência, é efetuado o produto das forças pelas coordenadas dos sensores, determinando-se, assim, a coordenada do CG humano.

Na terceira parte do programa é efetuada a soma dos dois resultados anteriores para a composição do CG resultante que será usado no controle da cadeira.

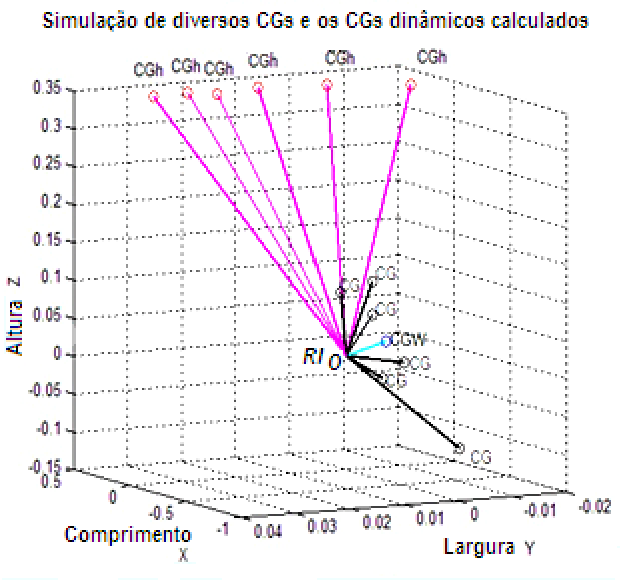


FIGURA 8 - Composição de diversos centros de massa humanos simulados (CGh) + CGw (fixo) gerando o CG resultante
FONTE: Dados da pesquisa.

Na terceira etapa, é efetuada a soma das coordenadas dos CGs humanos e da cadeira, obtendo-se, assim, o CG resultante. Todas operações podem ser vistas nos anexos.

4.2.1 Movimentos rotacionais

As rotações finitas são um dos aspectos da cinemática e, apesar da existência de diversas metodologias, a que foi empregada neste trabalho se baseia fundamentalmente em Newton e nos ângulos de Euler, uma das parametrizações mais encontradas na literatura, por apresentar uma visualização mais imediata do problema.

Apesar disso, essa metodologia pode levar a determinadas singularidades que devem ser evitadas ao longo do projeto, através de uma limitação dos movimentos.

No projeto dessa cadeira de rodas, as limitações podem ocorrer naturalmente, ou seja, as estruturas mecânicas nas bordas servem de batente, como nos projetos das cadeiras de rodas convencionais elétricas ou não. Mas tais limitações podem ocorrer de uma forma não natural, como é o caso do programa de controle limitar o movimento em função dos sensores.

No plano xz

Rotação em torno do eixo y:

$$R_{Y\theta} = \begin{bmatrix} \cos\theta & -sen\theta & 0 \\ sen\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

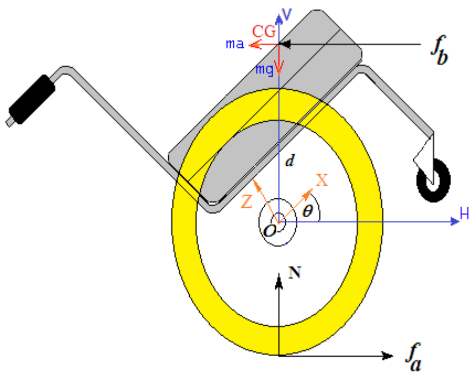


FIGURA 9 - Diagrama espacial da cadeira de rodas no plano XZ
FONTE: Dados da pesquisa.

No plano xy

Rotação em torno do eixo z:

$$R_{Z,\phi} = \begin{bmatrix} \cos\beta & 0 & sen\beta \\ 0 & 1 & 0 \\ -sen\beta & 0 & \cos\beta \end{bmatrix}$$

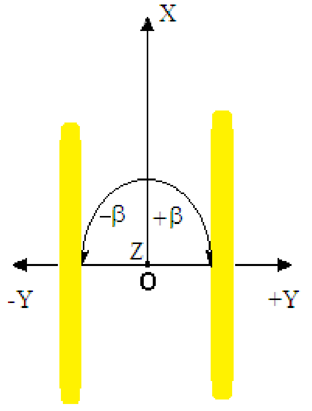


FIGURA 10 - Diagrama espacial da cadeira de rodas no plano XY
FONTE: Dados da pesquisa.

Convém lembrar que, conforme a movimentação que se faça nesse plano (em torno do eixo Z), pode, ou não, ocorrer anteriormente uma rotação no plano xy (cadeira em equilíbrio), pois, dessa forma, este movimento seria uma matriz produto das duas matrizes de rotação.

No plano zy

A cadeira pode entrar em um plano inclinado e, ao torcer este plano lateralmente, temos que considerar o movimento de forma inevitável, conforme se vê abaixo:

Rotação em torno do eixo x:

$$R_{X,\varphi} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -sen\phi \\ 0 & sen\phi & \cos\phi \end{bmatrix}$$

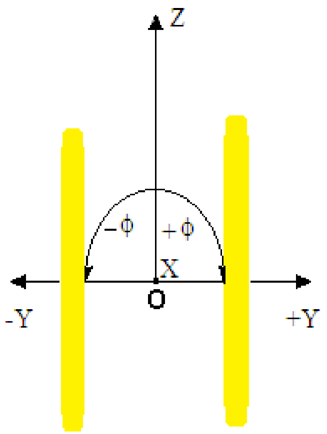


FIGURA 11 - Diagrama espacial da cadeira de rodas no plano ZY
FONTE: Dados da pesquisa.

Apesar de esta rotação ser possível (no plano inclinado lateral), colocou-se uma restrição inicial neste trabalho para tal movimento, de forma que as duas rodas da cadeira estejam sempre niveladas simultaneamente e essa transformação ou correção já não seria considerada.

4.2.2 Movimento translacional

As combinações de movimentos nesses planos, para incluir os movimentos de translação, tornam o modelo ainda mais complexo, conforme demonstrado a seguir.

No plano XZ

A consideração da translação nesse plano faz sentido quando a cadeira avança ou recua na direção do eixo X, no plano horizontal, e/ou sobe uma rampa em linha reta, sendo que, para este caso, o deslocamento do referencial local em relação ao inercial se faz sentir através de uma variação tanto de **x** como de **z**.

No plano YZ

A consideração nesse plano só faz sentido quando a cadeira sobe uma rampa com giro sobre esta para a esquerda ou para a direita e o deslocamento do referencial local em relação ao inercial se faz sentir tanto para y como para z. No caso deste estudo, considera-se que as rampas são planas e, mesmo no caso de giro, não existem deslizamentos que mudem sua posição.

No plano XY

Os movimentos nesse plano são mais ilimitados devido à composição que se pode fazer para atingir esses objetivos pré-determinados, só havendo restrição lateral para este tipo de translação. Considerando a FIG. 12 a seguir:

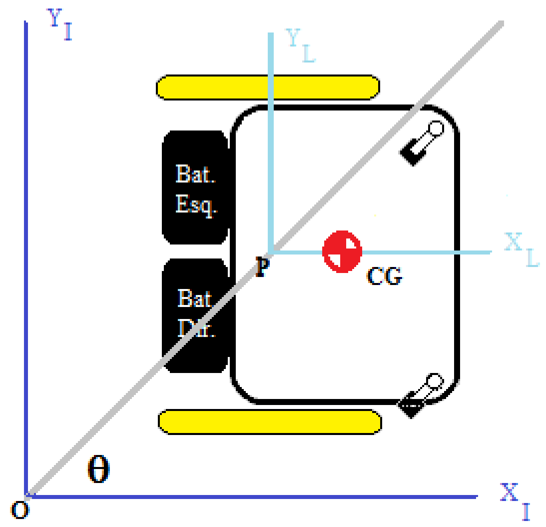


FIGURA 12 - Diagrama espacial da cadeira de rodas no plano XY
FONTE: Dados da pesquisa.

Considera-se que:

b = largura entre a roda e ***O*** o referencial inercial, no chassis da plataforma;
V$_e$ = a velocidade da roda esquerda = $\omega_e r_e$ (sendo que r_e é o raio da roda esquerda = r = constante);
V$_d$ = a velocidade da roda direita = $\omega_d r_d$ (sendo que rd é o raio da roda direita = r = constante) e;
R = a distância do ponto médio **I** do eixo ao **CIR** (Centro Instantâneo de Referência).

No plano ZY

Esse movimento não existe devido à restrição de não escorregamento lateral.

4.2.3 Centro de referência inercial (CIR)

Existe nos Apêndices um estudo sobre os diversos movimentos da cadeira, levando em consideração o centro de referência inercial.

4.3 Modelagem matemática da cadeira: equações do movimento

Na inclinação, a cadeira se comporta como:

4.3.1 O princípio do Pêndulo Invertido aplicado a este estudo

O pêndulo invertido, é um dos sistemas mecânicos mais utilizado no estudo de controle de sistemas instáveis, como é o caso desta cadeira robótica e equilibrista, no controle de lançamento de veículos espaciais e no controle da postura em seres humanos. O sistema consiste em um pêndulo invertido preso a um carrinho motorizado que se movimenta sobre um trilho. O objetivo do controle é manter o pêndulo equilibrado na posição vertical, mesmo quando perturbações são aplicadas ao sistema. Esse sistema ilustra as dificuldades práticas associadas com aplicações de sistemas de controle no mundo real, sendo, portanto, de interesse para os estudos em tecnologia de controle. O carro de massa **M** desliza sem atrito no eixo dos x. Um pêndulo de massa m e de comprimento d está fixo ao carro. Aplica-se uma força **F** ao carro de maneira a manter o pêndulo por cima do carro, segundo as notações:

Pêndulo: $d\theta/dt = \theta'$ e $d^2\theta/dt^2 = \theta''$ Carro: $dx/dt = x'$ e $d^2x/dt^2 = x''$

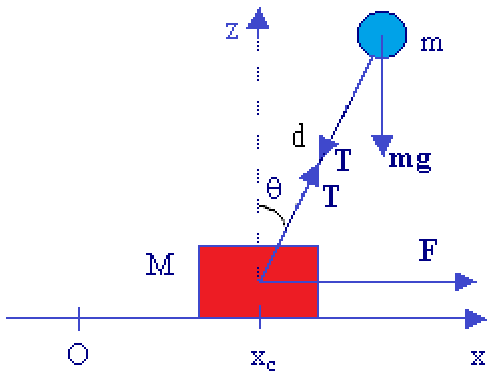


FIGURA 13 - Diagrama geométrico para modelagem do pêndulo na inclinação
FONTE: Dados da pesquisa.

4.3.2 Modelagem Matemática Newtoniana

O princípio dos trabalhos virtuais, consequência dos deslocamentos virtuais, utilizado no modelamento clássico, na ótica newtoniana, que apenas estabelece as condições de equilíbrio estático de sistemas, não pode ser aplicado em problemas dinâmicos de multicorpos, a não ser que fosse estendido através de outro princípio, o que mais tarde foi realizado por d'Alembert. Esse procedimento nos leva a montar as equações básicas que permitirão obter os dados ligados ao seu comportamento dinâmico, e que futuramente permitirão implementar o seu controle que será tão real quanto a realidade do modelo obtido.

Neste estudo obteremos o modelo matemático de sistemas mecânicos híbridos (ou seja, aqueles cujas massas executam movimentos de translação e rotação), a partir da aplicação da 2ª Lei de Newton e da Equação de Euler. Nosso estudo ficará restrito ao movimento de corpos rígidos no plano XZ, também conhecido simplesmente por movimento de inclinação. Uma grande maioria dos mecanismos existentes nos sistemas reais se enquadra nesse tipo de movimento, mas em uma cadeira robótica o modelamento é mais complexo.

$$(d^2\theta/dt^2) = ((M+m)\, g\, \text{sen}\theta - md(d\theta/dt)^2 \text{sen}\theta \cos\theta - F\cos\theta)/(M + m\, \text{sen}^2\theta)d \qquad \text{(eq.4.7)}$$

$$(d^2x/dt^2) = (m\, d(d\theta/dt)^2 \text{sen}\theta - m\, g\, \text{sen}\theta \cos\theta + F)/(M + m\, \text{sen}^2\theta) \qquad \text{(eq.4.8)}$$

A eficiência da formulação newtoniana é devida à sua natureza primitiva, que não eleva tanto o tempo de processamento, mantendo o custo em $\mathbf{O(n^2)}$[1], e não é tão maior quanto o número de corpos acoplados e seus graus de liberdade (Ver no Apêndice 2 - Conceitos).

4.3.3 Modelagem Matemática Lagrangeana

Como sistemas com vínculo não possuem deslocamentos virtuais independentes, Lagrange reescreveu o princípio de D'Alembert em termos de um número mínimo de coordenadas, ditas generalizadas, de forma que, os coeficientes dos deslocamentos virtuais podem ser anulados separadamente, obtendo-se um sistema de equações diferenciais ordinárias, denominadas equações de Lagrange. Uma formulação simples e sistemática.

Sendo as equações de Lagrange escalares, sua derivação é mais simples que a das equações de Newton. Adicionalmente, apenas os deslocamentos e velocidades precisam ser calculados, excluindo-se a aceleração. Além disso, as forças de reação que não produzem trabalho, não precisam ser incluídas na formulação das equações do movimento, como era feito para as equações de Newton. Para o caso da cadeira de rodas em questão, que possui um número de graus de liberdade elevado, escrever o lagrangeano de um dispositivo dessa natureza é uma tarefa bem árdua. Apenas poderia ser inviabilizada se o número de derivadas simbólicas fosse muito elevada, o que não será o caso, levando-se em conta as restrições que serão impostas, para processamento computacional. Outra forma de diminuir o custo do processamento pela formulação de Lagrange sera ignorar as forças devidas a Coriolis e centrífuga. Contudo, estas podem ser significativas quando o dispositivo, a cadeira, executa os movimentos em velocidades rápidas, principalmente visando ao equilíbrio.

A formulação de Lagrange fornece equações explícitas de estado e pode ser usada tanto para resolver problemas de dinâmica direta, ou seja, dados os torques/forças, as equações nos fornecem as acelerações que são, então, integradas (ver ODE) para retornar as coordenadas generalizadas e suas velocidades, como para a dinâmica inversa que, dadas as coordenadas generalizadas e suas duas primeiras derivadas no tempo (velocidades e acelerações), retornam as forças e torques generalizados.

A pouca eficiência da formulação lagrangeana é devida à sua natureza matricial, que eleva o tempo de processamento, elevando o custo $\mathbf{O(n^4)}$, e é tão maior quanto o número de corpos acoplados e seus graus de liberdade.

4.3.4 Modelo Lagrangeano

Partindo-se da formulação da energia cinética (K) e da energia potencial (P), obtém-se a função lagrangiana (L) da forma:

$$L = K - P \qquad \text{(eq.4.9)}$$

Obtém-se as mesmas equações como o desenvolvimento segundo Newton:

$$(d^2\theta/dt^2) = ((M+m)\ g\ \text{sen}\theta - md(d\theta/dt)^2\text{sen}\theta\cos\theta - F\cos\theta)/(M + m\ \text{sen}^2\theta)d \qquad \text{(eq.4.10)}$$

$$(d^2x/dt^2) = (m\ d(d\theta/dt)^2\text{sen}\theta - m\ g\ \text{sen}\theta\cos\theta + F)/(M + m\ \text{sen}^2\theta) \qquad \text{(eq.4.11)}$$

4.3.5 Modelo da cadeira segundo Newton

Em primeira aproximação, desacoplando os movimentos, como foi sugerido para efeitos de simplificação, pode-se interpretar o dispositivo geometricamente fazendo-se a composição das forças que nele atuam, segundo as orientações horizontal e vertical, obtendo-se as equações básicas de Newton em função das condições iniciais, ou seja :

$$F_H = \sum_{i=1}^{n} f_i = 0$$

$$F_V = \sum_{j=1}^{n} f_j = 0$$

Sabe-se que, ao estudar este movimento desacoplado, somado à natureza de interpretação estática dessa metodologia newtoniana, há a omissão das características dinâmicas que algumas vezes se mostram relevantes, dependendo da natureza do dispositivo, o que é válido para o efeito didático neste trabalho e na aplicação do controle ao dispositivo real, como será argumentado nas conclusões.

Para se modelar a cadeira, através das equações de Newton que descrevem um comportamento dinâmico simplificado do sistema, podem-se estabelecer mais algumas restrições e desacoplar os movimentos, assumindo inicialmente que a cadeira não executará determinados movimentos, como o da rotação (Roll), nem de deslocamentos dos movimentos de translação no plano **YZ**. Considera-se, ainda, como referência inercial na Terra o ponto ***I*** dos eixos cartesianos (X,Y,Z), que coincide no tempo inicial zero com o ponto ***O*** dos eixos cartesianos (i,j,k), no centro do eixo traseiro entre as rodas da cadeira e os ângulos **θ,** que é o ângulo que a cadeira de rodas faz no plano XZ (giro em torno do eixo Y), conforme a FIG. 13, e **β** que é o ângulo que a cadeira de rodas faz no plano XY (giro em torno do eixo Z),

conforme a FIG. 12. Assume-se que um controlador (Convencional e Não Convencional) a ser desenvolvido no capítulo 5 exerce uma força na cadeira de forma a impulsioná-la para o sentido positivo, conforme a figura a seguir, para o caso da inclinação e, ainda, para realizar o equilíbrio ou, então, uma sequência de inclinação e giro lateral, conforme a FIG. 4.

Conforme esta força (corrente que produz no motor DC, um torque), a cadeira pode vir a equilibrar ou simplesmente deslocar no plano horizontal seguindo em frente. Ver-se-á, mais adiante, no capítulo 5, que a equação não linear da dinâmica de ordem 3 obtida aqui será colocada sob a forma de equações lineares com suas respectivas variáveis de estado onde haverá diversas alternativas de deslocamento, como foi mencionado na introdução (giro à direita, à esquerda, rodar, inclinar e outros).

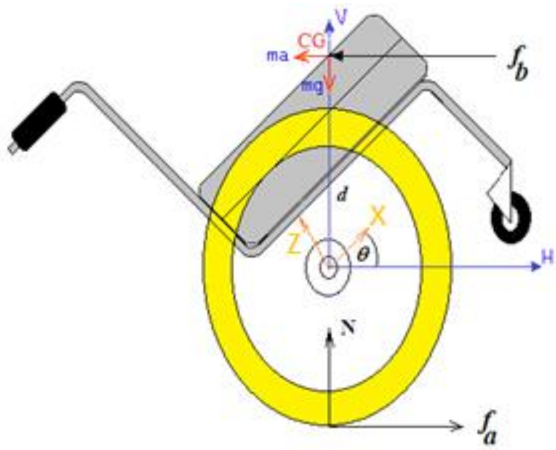


FIGURA 14 - Inclinação e equilíbrio
FONTE: Dados da pesquisa.

O ângulo de 0^0 (diferentemente do pêndulo invertido que nunca poderá ser elevado se ocupar este ângulo no momento da força aplicada inicial) deve ser a origem da posição do chassis da cadeira (no plano horizontal), que tem seu conjunto impulsor (motores) fixado no chassis, de forma que este gire até aproximadamente 22^0, sendo que a posição ocupada anteriormente pelo CG era de 68^0, em repouso, e 90^0 ,em equilíbrio, talvez esta uma inclinação confortável para o paciente.

Dentro dos procedimentos para se modelar um sistema dinâmico usando as leis da mecânica de Newton, tem-se o de analisar os movimentos separadamente, indicando todas as forças atuantes. Obtêm-se, então, um conjunto de equações tanto para os movimentos rotacionais como os translacionais.

4.3.6 Forças na horizontal

As forças horizontais atuantes na cadeira de rodas para o sistema impulsor são a força (no final são correntes, nos motores esquerdo e direito que se transformam em torques) aplicada pelo controlador **u** (na verdade u1 do motor esquerdo + u2 do motor direito)**,** a força de atrito da roda no solo, pois dispositivos mecânicos exploram a fricção $\mathbf{f_{at}}$ do contato com o solo para se locomoverem, e a força de reação **H.** Esse movimento de translação do centro de massa (em relação ao referencial inercial) do conjunto impulsor pode ser descrito pela equação:

$$M \frac{d^2 x}{dt^2} + f_{at} \frac{dx}{dt} + H = u(t) \qquad \text{(eq.4.12)}$$

Sendo que:

M = Massa do conjunto impulsor levada para o M_{CG};
f_{at} = Fricção no conjunto impulsor (lembrar que f_{at} = ccs sgn(dx/dt) + ccv (dx/dt));
H = Reação na horizontal;
u(t) = Ação ou força de impulsão devido aos dois motores elétricos.

Somando as forças atuantes no **CG** cuja haste imaginária **d** (não confundir nas equações com a derivada d) está ligada entre este ponto e a referência inercial **O** no centro do eixo traseiro, tem-se a equação para **H** da forma:

$$H = M \frac{d^2 x(t)}{dt^2} - M.d.\frac{d^2\theta(t)}{dt^2} sen\theta(t) - M.d.\left(\frac{d\theta(t)}{dt}\right)^2.\cos\theta(t) \qquad \text{(eq.4.13)}$$

Substituindo a equação 4.4.2 na equação 4.4.1 e considerando que a massa M do conjunto impulsor vai para a massa total concentrada que está no CG (M_{CG}), obtém-se a primeira equação do movimento, ou seja:

$$M_{CG}.\frac{d^2 x(t)}{dt^2} + f_{at}.\frac{dx(t)}{dt} - M_{CG}.d.\frac{d^2\theta(t)}{dt^2}.sen\theta(t) - M_{CG}.d.\left(\frac{d\theta(t)}{dt}\right)^2 \cos\theta(t) = u(t)$$

(eq.4.14)

4.3.7 Forças na vertical

Para obter a segunda equação do movimento, somam-se as forças atuantes na vertical do CG, que são a força peso **P** (em função da massa **m** do usuário e da força **g**), a força de reação horizontal **H,** decomposta segundo o eixo imaginário **d**, a força de reação vertical **N** (ou normal, sendo que H e N são as forças atuantes no acoplamento) da forma:

$$- N.\cos\theta(t) - H.sen\theta(t) - m.g.\cos\theta(t) = m.d.\frac{d^2\theta(t)}{dt^2} - m.\frac{d^2x(t)}{dt^2}.sen\theta(t) \qquad \text{(eq.4.15)}$$

É preciso resolver a equação acima para **N** e **H, o** que pode ser realizado por meio de uma manipulação algébrica simples, ou seja, multiplicar todos os termos da equação 4.4.4 por **d** e substituir nesta a equação 4.4.5 abaixo da soma dos momentos no CG:

$$N.d.\cos\theta(t) - H.d.\text{sen}\,\theta(t) + t_{at}.\frac{d\theta(t)}{dt} = J.\frac{d^2\theta(t)}{dt^2} \qquad \text{(eq.4.16)}$$

Sendo que:

J = É o momento de inércia do seu centro de gravidade;
d = Distância do CG à referência;
t_{at} = Fricção devido a torque no eixo das rodas (lembrar que t_{at} = ccs . sgn(dθ(t) /dt) + ccv.(dθ(t) /dt)).

Considerando a massa **m** do usuário levada ao CG (M_{CG}), obtém-se a segunda equação do movimento:

$$- - MCG.g.d.\cos\theta(t) + MCG.d.\frac{d^2x(t)}{dt^2}.\text{sen}\,\theta(t) + t_{at}.\frac{d\theta(t)}{dt} = \left(J + MCG.d^2\right).\frac{d^2\theta(t)}{dt^2} \qquad \text{(eq.4.17)}$$

Pode-se linearizar as equações de movimento em torno de um ponto de trabalho estacionário (ângulo de $\theta = 90^0 + \gamma$), considerando o CG para pequenos giros de um ângulo γ (bem pequeno) sendo que sen θ = 1; cos θ = γ e $(d\theta/dt)^2 = 0$ para estabelecer um modelo linear de referência a ser utilizado futuramente no controle inverso:

$$MCG.\frac{d^2x(t)}{dt^2} + f_{at}\frac{dx(t)}{dt} - MCG.d.\frac{d^2\gamma(t)}{dt^2} = \mathrm{u(t)} \qquad \text{(eq.4.18)}$$

$$\left(J + MCG.d^2\right).\frac{d^2\gamma(t)}{dt^2} + MCG.g.d.\gamma(t) - \mathrm{t}_{at}\frac{\mathrm{d}\gamma(t)}{\mathrm{dt}} = MCG.d.\frac{d^2x(t)}{dt^2} \qquad \text{(eq.4.19)}$$

Aplicando Laplace nas equações anteriores, a relação entre elas leva a uma função de transferência da forma:

$$\frac{\gamma(s)}{U(s)} = \frac{MCG.d.s}{MCG.J.\mathrm{s}^3 + \left(MCG.\mathrm{d}^2.\mathrm{f}_{at} + \mathrm{f}_{at}.J - MCG.\mathrm{t}_{at}\right).\mathrm{s}^2 + \left(\mathrm{MCG}^2.g.d - \mathrm{f}_{at}.\mathrm{t}_{at}\right).s + MCG.g.d.\mathrm{f}_{at}} \qquad \text{(eq.4.20)}$$

Obs.: U(t) = É, como dissemos anteriormente, a força ou ação resultante [$\pm$]Ud + [$\pm$]Ue devida aos dois motores com seus respectivos sinais.

No plano a cadeira se comporta como um robô.

Nesta seção é apresentado um modelo de um robô móvel semelhante à cadeira de rodas de duas rodas diferenciais. A metodologia seguida é idêntica àquela usada em outros tipos de veículos, tais como os robôs móveis terrestres de tração convencional (tipo automóvel) (KROGH; GRAETTINGER,1989; PAPPAS; KYRIAKOPOULOS, 1992; KROGH; FENG, 1991), de tração síncrona (ZHAO; BEMENT, 1992), ou com rodas omnidirecionais (MUIR; NEUMAN, 1987).

A escolha de um veículo de tração diferencial é utilizada neste trabalho por motivos de ordem prática. Na verdade, a cadeira de rodas desenvolvida em laboratório é desse tipo e os estudos de um caso particular nos permitem obter um conhecimento detalhado de grande utilidade sem perda de generalidade.

Apresentam-se, neste capítulo, as equações da cinemática e da dinâmica da cadeira de rodas, dos motores, assim como a do atrito. Finalmente, apresentamos o modelo dinâmico global para diferentes situações, no que diz respeito às posições desejadas, ao atrito nas rodas sem escorregamento, só com o escorregamento na tração e com o escorregamento.

O estudo, tendo em vista a obtenção de um modelo bastante detalhado da cadeira, tem duas motivações: por um lado, do modelo mais preciso podem ser retirados modelos mais simples que serão utilizados na identificação e no controle da cadeira. Por outro lado, o conhecimento detalhado do comportamento da cadeira robótica, permite a determinação da

ordem de grandeza (e, portanto, o limite superior) do efeito dos desvios da linearidade não considerados nos modelos mais simples.

4.4 Cinemática

4.4.1 Sistemas não holonômicos

A cadeira pode ser considerada como tendo duas rodas motrizes traseiras independentes e fixas e duas rodas giratórias livres frontais (“castors”), para garantir estabilidade, que permitem, no plano, três graus de liberdade: posição no plano xy e ângulo β de orientação. Além disso, ela pode inclinar em relação a este plano, adquirindo um quarto grau de liberdade, ou seja, ângulo θ de inclinação em relação ao solo em que está.

A cadeira robótica móvel aqui considerada está sujeita a restrições não holonômicas. Essa característica torna o problema de controle mais difícil de resolver.

Um sistema dinâmico é geralmente caracterizado por um conjunto de equações diferenciais. No entanto é usual existirem restrições no estado desse sistema, como por exemplo, restrições no espaço de estados admissíveis. Se estas restrições envolvem apenas os parâmetros de configuração do sistema (equações que envolvem apenas as coordenadas ou posições em um sistema mecânico), são restrições holonômicas.

As restrições não holonômicas surgem quando temos restrições envolvendo as derivadas dos parâmetros de configuração (o espaço de configuração é o subespaço de todas as possíveis "posições" não apenas desta cadeira, mas de qualquer robô (Laumood, 1993) com estas características, e neste caso *x*, y, z, θ e β) e, como tal, não limitam diretamente o espaço de configurações possíveis. São traduzidas por equações não-integráveis envolvendo as derivadas dos parâmetros de configuração. No caso desta cadeira, as restrições não holonômicas incluem, por exemplo, as condições de não-escorregamento que não limitam diretamente o conjunto de configurações possível. De fato, embora a cadeira possa teóricamente estar em qualquer ponto do plano com qualquer orientação, não pode passar diretamente de uma configuração arbitrária para outra sem efetuar manobras. Por exemplo: não pode se mover lateralmente.

Pode-se dizer também que num certo sentido ela é globalmente controlável, mas não localmente. O fato de existirem restrições não holonômicas faz com que não se possa “a priori” restringir o espaço de estados de forma a eliminar essas restrições. Isso implica que tais restrições têm que ser sempre consideradas, quando se está resolvendo o problema.

4.4.2 Cinemática do centro das rodas em função da referência inercial *O*

Na análise cinemática e dinâmica, consideram-se dois referenciais, um inercial e fixo ($X_I Y_I$), e outro, móvel, fixo no robô (***ij***, um referencial vetorial). Para a origem deste último, considera-se o ponto ***O*** no meio do eixo imaginário de tração do robô. O eixo ***i*** está na direção do comprimento do robô (direcionado para frente) e o eixo ***j*** está na transversal (perpendicular a ***i***).

Antes de se escrever as equações cinemáticas da cadeira, demonstram-se as equações cinemáticas dos centros das rodas em função do ponto ***O*** (ver Figura 15) e da orientação destas.

Isso é feito para que posteriormente se possa escrever as equações da dinâmica para as rodas.

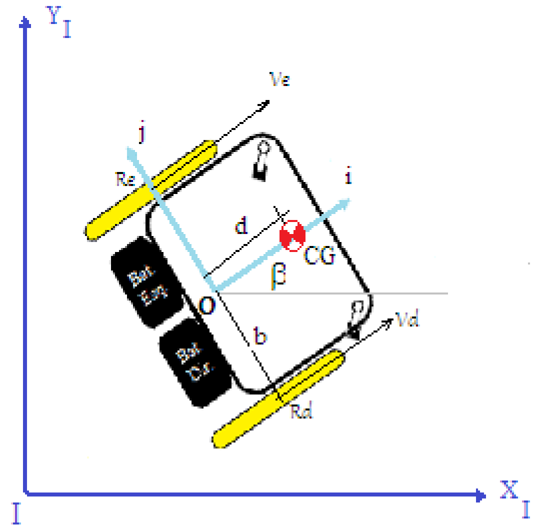


FIGURA 15 - Diagrama de corpo livre da cadeira
FONTE: Dados da pesquisa.

De acordo com a FIG. 15 anterior, pode-se escrever para o ponto $\boldsymbol{R_d}$:

$$x_{Rd} = x_0 + b \sin\beta \quad \text{(eq.4.12)}$$
$$y_{Rd} = y_0 - b \cos\beta \quad \text{(eq.4.13)}$$

Derivando-se as equações (4.3.1) e (4.3.2), obtém-se a velocidade de $\boldsymbol{R_d}$ em função da velocidade de ***O*** e da velocidade angular:

$$V_{dx} = \dot{x}_{Rd} = \dot{x}_0 + b \cos\beta\,\dot{\beta} \quad \text{(eq.4.14)}$$
$$V_{dy} = \dot{y}_{Rd} = \dot{y}_0 + b \sin\beta\,\dot{\beta} \quad \text{(eq.4.15)}$$

Derivando-se mais uma vez, obtém-se a aceleração de $\boldsymbol{R_d}$ em função da aceleração angular e da referência inercial ***O*** :

$$\alpha_{dx} = \ddot{x}_{Rd} = \ddot{x}_0 + b\cos\beta\,\ddot{\beta} - b\sin\beta\,\dot{\beta}^2 \qquad \text{(eq.4.16)}$$
$$\alpha_{dy} = \ddot{y}_{Rd} = \ddot{y}_0 + b\sin\beta\,\ddot{\beta} + b\cos\beta\,\dot{\beta}^2 \qquad \text{(eq.4.17)}$$

Da mesma forma, é possível obter a aceleração do ponto $\boldsymbol{R_e}$ em função da aceleração angular e da referência inercial ***O***:

$$\alpha_{ex} = \ddot{x}_{Re} = \ddot{x}_0 - b\cos\beta\,\ddot{\beta} - b\sin\beta\,\dot{\beta}^2 \qquad \text{(eq.4.18)}$$
$$\alpha_{ey} = \ddot{y}_{Re} = \ddot{y}_0 - b\sin\beta\,\ddot{\beta} - b\cos\beta\,\dot{\beta}^2 \qquad \text{(eq.4.19)}$$

4.4.3 Modelo cinemático da cadeira de rodas

Admitindo-se que $\boldsymbol{V_d}$ e $\boldsymbol{V_e}$ são as velocidades de $\boldsymbol{R_d}$ e $\boldsymbol{R_e}$, respectivamente, pode-se escrever as velocidades lineares e a velocidade angular de ***O*** em função daquelas velocidades, assim:

$$V_{xO} = \dot{x}_O = \frac{V_{dx}+V_{ex}}{2}\cos\beta \qquad \text{(eq.4.20)}$$

$$V_{yO} = \dot{y}_O = \frac{V_{dx}+V_{ex}}{2}\sin\beta \qquad \text{(eq.4.21)}$$

$$\omega_O = \dot{\beta}_O = \frac{V_{dx}-V_{ex}}{2} \qquad \text{(eq.4.22)}$$

Admitindo-se que não há escorregamento, então, as velocidades $\boldsymbol{V_d}$ e $\boldsymbol{V_e}$ são as velocidades lineares das rodas e se relacionam às velocidades angulares através do raio **r** das rodas. Usualmente, o valor dessas velocidades é usado para controle. Esse modelo está na forma:

$$\vartheta = f(\boldsymbol{V},\boldsymbol{U}) \qquad \text{(eq.4.23)}$$

Sendo que:

O vetor de estado ***V*** é constituído por x, y e β e o vetor de controle ***U***, por $\boldsymbol{V_d}$ e $\boldsymbol{V_e}$.

Esse é o modelo mais utilizado (MUIR; NEUMAN, 1987; KROGH; GRAETTINGER, 1989; TURENNOUT *et al.*, 1992) para o controle desse tipo de cadeira de rodas.

4.5 Dinâmica

Um aspecto importante a se considerar na modelagem da cadeira é o da descrição da sua dinâmica. Esse estudo é fundamental para que se possa englobar efeitos dinâmicos no controle, como, por exemplo, a inércia ou a respostas dos motores. Para a realização de tal estudo, é necessário considerar as equações da dinâmica dos motores na medida em que estas determinam os torques que surgem nas equações, que geram as forças, e, consequentemente, no comportamento da cadeira.

4.5.1 Motores

Os motores utilizados neste caso são os de corrente contínua, com imãs permanentes e suas redutoras. Considerou-se, como variável de controle, a tensão aplicada ao motor. Essa tensão é o valor médio da forma de onda quadrada gerada pelo conversor *Pulse Width Modulation* (PWM), que controla o motor. Os diagramas da forma de tensão estão conforme a FIG. 16.

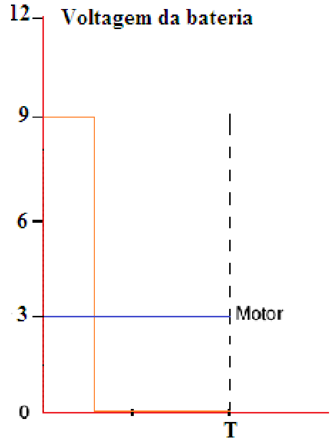


FIGURA 16 - Controlador gerador do PWM e velocidades
FONTE: Dados da pesquisa.

Calcula-se, através do programa dos controladores ("Software"), um valor que traduz um código recebido do supervisor que, uma vez aplicado a um acionador ("driver") de

potência, produz uma tensão resultante do conversor PWM que controla os motores. A referência é um código de valor binário que produz cinco níveis de velocidade entre 0000h e FFFFh. Tal valor recebido pelo programa do controlador gera o PWM de forma a fazer variar o seu "duty-cycle". Esta é a forma de aplicar a energia correta fornecida ao motor, resultando, na saída, em uma tensão média entre 0V e +12V (a ação intermediária ***U*** sob a forma de tensão). A variação do "duty-cycle" traduz-se, na prática, por uma variação do valor médio da tensão que é aplicada ao motor, isto porque este se comporta como um filtro passa-baixo (DEWAN; STRAUGHEN, 1984).

Na FIG. 17, pode-se observar um diagrama de blocos do esquema equivalente do motor:

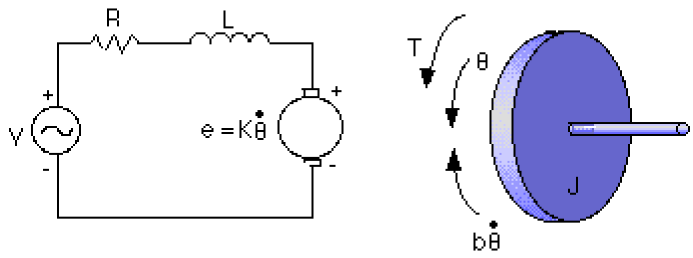


FIGURA 17 - Diagrama de blocos do esquema equivalente do motor
FONTE: CARNEGIE-MELLON (2001).

O torque produzido pelo motor é diretamente proporcional à corrente do seu induzido.

$$T = k_t i_a \qquad \text{(eq.4.24)}$$

Pelo fato de o motor se encontrar em movimento, o campo magnético do indutor induz uma tensão (força contraeletromotriz). A tensão induzida pelo movimento seria uma força que se opõe ao movimento do motor, daí a sua denominação. O seu valor varia linearmente com a velocidade do motor.

$$e_a = k_e i_a \qquad \text{(eq.4.25)}$$

Neste esquema, desprezou-se a indutância **L** do motor, eliminando-se, dessa forma, uma componente da variável de estado. Essa aproximação é usual (COSTA, 1995;

KLAFTER; NEGIN, 1989), uma vez que o efeito da indutância é a introdução de um polo em altas frequências. Como este não é dominante, tem pouca influência no comportamento do sistema que é um passa-baixo (DEWAN; STRAUGHEN, 1984; KLAFTER; NEGIN, 1989).

Pela aná1ise do esquema anterior, vê-se que:

$$V_a = R_a i_a + e_a \quad \text{(eq.4.26)}$$

É usual considerar:

$$k_t = k_e = k \quad \text{(eq.4.27)}$$

O que equivale, em termos, a menos das perdas elétricas no motor a **potência mecânica** igual à **potência elétrica**. Desprezando somente as perdas devidas à indução, tem-se, pois:

$$T\omega = V_a i_a - R_a i_a^2 \quad \text{(eq.4.28)}$$

A partir do que se pode encontrar (eq.4.29). Substituindo (eq.4.26) em (eq.4.28) e atendendo a (eq.4.16), obtém-se:

$$V_a = \frac{R_a}{k} T + k\omega \quad \text{(eq.4.29)}$$

Ou

$$T = k_1\omega + k_2 V_a \quad \text{(eq.4.30)}$$

Com:

$$k_1 = -\frac{k^2}{R_a} \quad \text{(eq.4.31)}$$

$$k_2 = \frac{k}{R_a} \quad \text{(eq.4.32)}$$

É possível observar, a partir equações apresentadas, que existe urna relação linear entre a velocidade angular do motor **ω** e o torque **T** que ele pode fornecer. Essa relação é apresentada de forma gráfica na FIG. 18.

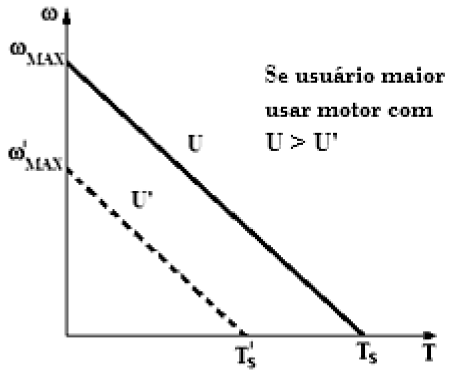


FIGURA 18 - Diagrama de velocidade-torque
FONTE: Dados da pesquisa.

Quando o motor está sem carga, a sua velocidade é máxima $\boldsymbol{\omega}_{\mathbf{max}}$. Por outro lado, quando o motor é bloqueado ("**s**talled", $\boldsymbol{\omega}$=0), o seu torque é máximo $\mathbf{T_s}$. Para diferentes valores de tensão **U** e **U'**, as retas que definem a relação entre **T** e $\boldsymbol{\omega}$ possuem uma inclinação igual, mas possuem diferentes ordenadas na origem. Para uma tensão menor **U'**, ambos os valores da velocidade e do torque máximos são menores.

No caso da cadeira robótica em estudo, o torque de carga não é constante, embora se atue na tensão do motor. O torque fornecido por cada um dos motores depende não só da dinâmica da cadeira, que determina a carga para os motores, mas também da velocidade angular $\boldsymbol{\omega}$ dos motores que pode ser considerada como uma variável de estado.

4.5.2 Dinâmica do chassis

Considera-se como ponte de interesse para o estudo da dinâmica da cadeira o ponto de referência inercial ***O***.

Pretende-se que este ponto ***O*** percorra uma dada trajetória no espaço.

Na FIG. 19, pode-se observar as forças que atuam na cadeira.

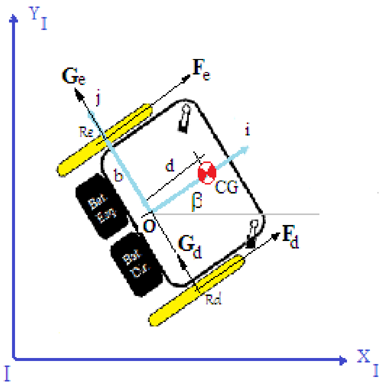


FIGURA 19 - Forças que atuam na cadeira
FONTE: Dados da pesquisa.

Limitando-se ao movimento no plano e, como tal, desprezando os momentos de giro ("roll"), consideram-se as forças $\mathbf{F_x}$ e $\mathbf{G_x}$, que atuam no eixo das rodas cuja largura é nula. $\mathbf{F_x}$ e $\mathbf{G_x}$ resultam da decomposição da força de atrito exercida nas rodas e, consequentemente no chassis da cadeira, nas componentes tangencial e normal à trajetória.

As forças **Fx** são as componentes tangenciais nas rodas e, como tal, responsáveis pela tração, ou seja, pela aceleração da cadeira.

As forças **Gx** são as componentes normais causadas pela aceleração centrípeta, devidas à variação da direção da velocidade da cadeira no tempo. O que acontece quando a cadeira descreve uma curva.

Sendo **x** e **y** as coordenadas cartesianas do ponto referencial inercial ***O*** (referencial XY) e **β** como a orientação da cadeira conforme a FIG. 19, pode-se escrever a partir da lei de Newton:

$$Ma_x = M\ddot{x} = (F_d + F_e)\cos\beta - (G_d + G_e)\sin\beta \qquad \text{(eq.4.33)}$$

$$Ma_y = M\ddot{y} = (F_d + F_e)\sin\beta + (G_d + G_e)\cos\beta \qquad \text{(eq.4.34)}$$

Se, em alternativa, considera-se a velocidade ***V*** linear da cadeira, tem-se, na mesma a lei de Newton, sob a forma:

$$Ma = M\dot{V} = F_d + F_e \qquad \text{(eq.4.35)}$$

A equação dos momentos para o ponto ***O*** é:

$$I_O\ddot{\beta} = (F_d - F_e)b \qquad \text{(eq.4.36)}$$

Sendo que ***Io*** pode ser obtido (GOLDSTEIN, 1980) a partir do momento de inércia em relação ao eixo vertical que passa pelo centro de massa.

$$I_O = I_{CG} + Md^2 \quad \text{(eq.4.37)}$$

Obtêm-se agora mais duas equações através da aplicação das leis de Newton ao movimento das rodas. Para cada roda, é possível escrever uma equação que relaciona o torque do motor a sua aceleração angular. Na FIG. 20, observa-se um diagrama das forças que atuam em cada roda:

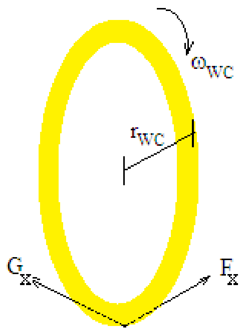


FIGURA 20 - Forças em uma roda em movimento
FONTE: Dados da pesquisa.

A força G_x, por ter a mesma direção da velocidade angular, é normal ao plano definido pela roda e não contribui para a sua aceleração angular. A velocidade angular da roda é $\boldsymbol{\omega_{wc}}$, $\mathbf{r_{wc}}$ *é* o seu raio, e $\mathbf{T_w}$ é o torque fornecido pelo motor. Para a roda, escreve-se, então:

$$I_{wc}\dot{\omega}_{wc} = T - Fr_{wc} \quad \text{(eq.4.38)}$$

Tal como visto anteriormente, o torque $\mathbf{T_w}$ *é* dado por (4.24), dado a partir do qual, pode-se escrever:

$$I_{wc}\alpha_d = I_{wc}\dot{\omega}_d = k_{d1}\omega_d + k_{e1}U_d - F_d r_{wc} \quad \text{(eq.4.39)}$$

$$I_{wc}\alpha_e = I_{wc}\dot{\omega}_e = k_{d2}\omega_e + k_{e2}U_e - F_e r_{wc} \quad \text{(eq.4.40)}$$

Sendo que:

ω_d, ω_e e r_{wc} são as velocidades angulares e os raios das rodas direita e esquerda, respectivamente. Assume-se que o momento de inércia I_{wc} de cada roda em relação ao seu eixo de rotação é igual para as duas rodas, sendo que as constantes k_{d1}, k_{e1}, k_{d2} e k_{e2} são dadas a partir de (4.31) e (4.32), dependendo apenas das características dos motores.

Uma vez que as forças **Gx** são derivadas da aceleração centrípeta, escreve-se:

$$G_d + G_e = F_c \qquad \text{(eq.4.41)}$$

A força centrípeta Fc pode ser dada em função do módulo da velocidade da cadeira e do raio de curvatura descrito ao percorrer a trajetória, conforme escrito abaixo:

$$F_c = \frac{MV^2}{R_c} \qquad \text{(eq.4.42)}$$

Sendo que **M** é a massa total da cadeira e **Rc** é o raio de curvatura instantâneo da trajetória.

A velocidade linear da cadeira, ***V***, é dada por:

$$V^2 = \dot{x}^2 + \dot{y}^2 \qquad \text{(eq.4.43)}$$

Admitindo-se que há duas rodas de largura nula e a inexistência de escorregamento, calcula-se ***R*** cinemáticamente por:

$$R_c = \mp\sqrt{\frac{\dot{x}^2+\dot{y}^2}{\dot{\beta}^2}} \qquad \text{(eq.4.44)}$$

O sinal de R_c é determinado como sendo positivo para curvas em que β é crescente e negativo para curvas em que β é decrescente. Note-se que R_c pode ser dado por:

$$R_c = \frac{V}{\dot{\beta}} \qquad \text{(eq.4.45)}$$

4.5.3 Rodas livres

Nos robôs com direção diferencial, normalmente existem rodas livres que

fornecem pontos de apoio adicionais. Gera1mente existem duas rodas motorizadas independentes, responsáveis quer pela tração, quer pela direção. As rodas livres que proporcionam os pontos de apoio são designadas de castores (“Castors”).

Quando se faz a análise do sistema, é usual considerar que os castores são perfeitos, isto é, são meramente pontos de apoio que possuem atrito nulo com o solo em todas as direções. Infelizmente, na realidade, isso não acontece. As rodas livres, além de oferecerem resistência no seu sentido de rolamento (atrito viscoso no seu eixo de rotação), não estão sempre alinhadas com o sentido do movimento desejado e nem se alinham instantaneamente. Quando do início de um dado movimento, o não alinhamento das rodas livres poderá introduzir uma perturbação muito significativa.

Uma das principais dificuldades a ser considerada e que aparece na análise do sistema é que os castores possuem um caráter aleatório. De fato, em duas situações idênticas, pode haver perturbações diferentes, correspondendo a diferentes alinhamentos das rodas. Esse caráter aleatório deve-se a características do atrito e do solo, cuja superfície é caracterizada por pequenas irregularidades.

Uma vez que a dependência da dinâmica das rodas livres e a sua influência na dinâmica global pelas características locais do solo não é passível de ser definida por um modelo com ruído conhecido, torna-se difícil a sua caracterização.

Um fato positivo, no que diz respeito aos castores, é que estes tendem a se alinhar com o movimento pretendido, sendo a perturbação introduzida de caráter apenas transitório. Isto é, o sistema constituído pelas rodas livres é estável e, com o tempo tende sempre a um ponto de equilíbrio.

Quando se pretende incluir, de alguma forma, os efeitos das rodas castores em um modelo de cadeira em que se assume um rolamento perfeito, encontra-se a dificuldade de, quando os castores estão desalinhados, as restrições de compatibilidade de rolamento serem violadas. Eventualmente, as rodas de tração deixam de rolar perfeitamente, o que viola algumas hipóteses subjacentes ao modelo, invalidando-o.

Nesta seção, tentar-se-á caracterizar, embora de forma bastante aproximada, as rodas livres. O objetivo disso é obter algum conhecimento qualitativo sobre o comportamento das rodas, considerando-se possíveis efeitos delas e, ainda, integrar tal conhecimento para se evitar situações de grande perturbação causada por elas.

Na FIG. 21, observa-se a cadeira em estudo com duas rodas livres (Steers ou Castors) colocadas na frente.

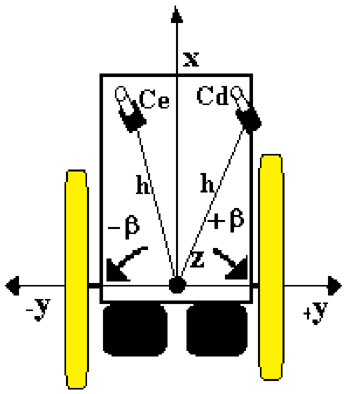


FIGURA 21 - Localização das rodas livres ("Castors")
FONTE: Dados da pesquisa.

As rodas livres giram em torno de um eixo vertical que não passa pelo seu centro de rotação (assinalado na figura pela letra O). Desse modo, os pontos de contato de cada roda com o solo (C_1 e C_2) podem descrever uma circunferência em torno desse eixo vertical (com um raio aproximadamente igual ao raio da roda). Uma vez que a variação dos pontos de contato com o ângulo da roda é, de fato, pequena, pois o raio é pequeno, causando pouca variação na distância h e no ângulo que esta apresenta em relação aos eixos da cadeira, consideram-se os pontos C_1 e C_2 fixos no eixo vertical de rotação das rodas.

4.5.4 Modelo dinâmico

Apresenta-se, aqui, um modelo dinâmico dos castores em que se considera a dinâmica de cada roda livre (dinâmica de rotação em torno do eixo vertical), traduzindo-se os efeitos da roda por uma força aplicada em cada ponto de contato com o solo. Força esta que altera a dinâmica de ***v*** e de **θ**.

Considera-se que a força de oposição ao movimento exercida em cada roda livre está na mesma direção da velocidade linear do ponto de contacto (***v*** que não depende da orientação da roda livre), conforme a FIG. 22 seguinte:

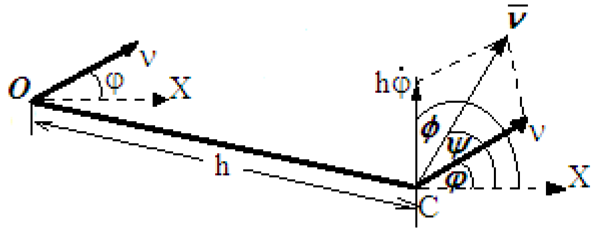


FIGURA 22 - Velocidades do ponto ***O*** e do ponto de apoio das rodas livres
FONTE: Dados da pesquisa.

Φ é o ângulo entre o eixo ***xx′*** e a componente da velocidade do ponto de apoio da roda livre (ponto ***C***), devido à rotação da cadeira (ou seja, o braço vezes a velocidade angular ***hφ'***). ***Ψ*** é o ângulo entre a velocidade linear do ponto **C** e o eixo dos ***xx′***. Essa velocidade é dada pela soma vetorial da velocidade linear da cadeira no ponto ***O*** (transladada para o ponto ***C***) com a componente, devido à rotação da cadeira (***hφ'***). Escreve-se, então, $\tilde{\boldsymbol{v}}$ em função de ***v*** e de ***θ***:

$$\tilde{v} = \frac{v\cos\varphi + h\dot{\varphi}\cos\phi}{\cos\Psi} \qquad \text{(eq.4.46)}$$

$$\Psi = arc\tan\left(\frac{v\sin\varphi + h\dot{\varphi}\sin(\phi+\varphi)}{v\cos\varphi + h\dot{\varphi}\cos(\phi+\varphi)}\right) \qquad \text{(eq.4.47)}$$

Em cada roda livre, considera-se o ângulo entre a direção da força que a roda exerce na cadeira e a direção longitudinal da roda (ζ , ver FIG. 23 seguinte):

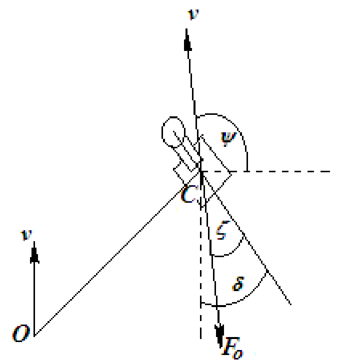


FIGURA 23 - Roda livre
FONTE: Dados da pesquisa.

$\boldsymbol{\delta}$ é o ângulo entre a direção da roda e o eixo $\boldsymbol{xx'}$ da cadeira (medido em relação ao semieixo negativo). $\boldsymbol{F_o}$ é a força que se opõe ao movimento da cadeira e que está direcionada no sentido oposto ao da velocidade do ponto de apoio $\boldsymbol{C}$. Esse ângulo traduz a posição da roda livre e pode ser dado por:

$$\delta = \zeta - \varphi + \Psi \quad \text{(eq.4.48)}$$

Nesse modelo, supõe-se que a força exercida pela roda livre na cadeira depende apenas do ângulo de alinhamento (ζ). Isso não é totalmente verdade, mas exprime a noção de que quanto mais a roda está alinhada, menor é a força que se exerce nela. Por outro lado, vamos considerar que a dinâmica das rodas livres é de primeira ordem. Mais uma vez, tem-se é uma simplificação, pois, de fato, as rodas livres alinham por ação de forças e momentos a que estão sujeitas, o que, no mínimo, sugere um sistema de segunda ordem. Essa simplificação é feita tendo em vista que os erros introduzidos pelos fatores aleatórios (características do solo, dinâmica não modelada, modelo do atrito e outros) são de tal forma relevantes que, face a eles, não se ganha precisão ao melhorar o modelo da dinâmica.

A dinâmica das rodas pode ser dada por um modelo como este:

$$\dot{\zeta} = -(k_{r1}\tilde{v} + k_{r2}) \sin \zeta \quad \text{(eq.4.49)}$$

As constantes $\boldsymbol{k_{rl}}$ e $\boldsymbol{k_{r2}}$ são determinadas experimentalmente. Note-se que a derivada de $\boldsymbol{\zeta}$ é proporcional ao seno deste angulo, o que implica que ζ tende para zero com o tempo. O que corresponde à situação de as rodas alinharem com o decorrer do movimento.

Para se modelar a força de oposição que os castores exercem sobre a cadeira, considera-se que esta é dada pelo seno de $\boldsymbol{\zeta}$ multiplicado por uma constante. O que significa

que a força será máxima para um ângulo de 90 graus e mínima para 0 graus. Ou seja, quando a roda se encontra colocada perpendicularmente em relação à direção de movimento, a força de oposição, causada pelo atrito, é maior, com ângulos diferentes, sendo que o castor em parte roda e em parte escorrega. Assim, a força de oposição não é tão elevada. Isto é:

$$F_o = F_{oMax} \sin \zeta \quad \text{(eq.4.50)}$$

Os efeitos da força de perturbação serão traduzidos por dois termos que afetarão a dinâmica das velocidades da cadeira (equações de ***v'*** e ***β''***). Assim, considerando iguais os dois motores, ter-se-á no modelo global:

$$a = \dot{v} = a_{11}v + b_{11}U_d + b_{12}U_e - F_{ov} \quad \text{(eq.4.51)}$$

$$\alpha = \dot{\omega} = a_{22}\omega + b_{21}U_d + b_{22}U_e - hF_{o\omega} \quad \text{(eq.4.52)}$$

As componentes $\boldsymbol{F_{ov}}$ e $\boldsymbol{F_{o\omega}}$ são dadas por:

$$F_{ov} = F_{oMax} \sin \zeta_1 \cos(\alpha_1 - \zeta_1) + F_{oMax} \sin \zeta_2 \cos(\alpha_2 - \zeta_2) \quad \text{(eq.4.53)}$$

$$F_{o\omega} = F_{oMax} \sin \zeta_1 \cos\left(\frac{\pi}{2} + \alpha_1 - \zeta_1 - \phi_1\right) + F_{oMax} \sin \zeta_1 \cos\left(\frac{\pi}{2} + \alpha_2 + \zeta_2 - \phi_2\right)$$

(eq.4.54)

Além das variáveis de estado já existentes, acrescentam-se mais duas, uma por cada roda livre. A dinâmica destas é dada por (eq.4.45) e (eq.4.46).

$$\dot{\zeta}_1 = -(k_{r11}\tilde{v} + k_{r12}) \sin \zeta_1 \quad \text{(eq.4.45)}$$

$$\dot{\zeta}_2 = -(k_{r21}\tilde{v} + k_{r22}) \sin \zeta_2 \quad \text{(eq.4.46)}$$

Obs.: Um fato que deve ser salientado é que quando $\boldsymbol{\zeta} = \pm\pi$, está-se diante de uma situação de equilíbrio instável. Realmente, em tal situação, é impossível prever "a priori" para qual dos lados a roda livre vira. A roda livre girará em torno do eixo vertical, em um sentido que dependerá de fatores aleatórios em jogo.

Considera-se que as rodas livres constituem uma incerteza do sistema, a incluir no modelo de forma multiplicativa, na saída (CHIANG; SAFODOV, 1992; MACIEJOWSKY, 1989), conforme a FIG. 24:

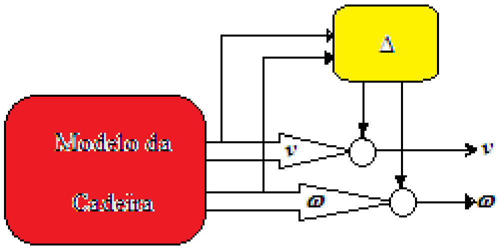


FIGURA 24 - Modelo com perturbações multiplicativas
FONTE: Dados da pesquisa.

4.6 Espaço de estados

A partir das equações diferenciais obtidas nos itens anteriores (malha aberta), foi obtido o modelo do sistema em Espaço de Estados (matrizes A, B, C e D). Este modelo foi adotado para efeito com o controlador "Fuzzy" adiante.

Existem dois procedimentos executados para se obter as matrizes do espaço de estado: o primeiro deles, conforme o item a seguir, é o de se utilizar do comando do Matlab TF2SS(GS), fazendo-se a conversão direta da Função de Transferência do Sistema para o Espaço de Estado, e o outro é o desenvolvimento exposto na apresentação anexa, através de uma metodologia a partir da qual se levam em consideração os elementos armazenadores de energia e as equações da dinâmica associadas a estes. A seguir, as matrizes do Espaço de Estado para o pêndulo adaptadas para a cadeira.

$$\begin{bmatrix}\dot{x}\\ \ddot{x}\\ \dot{\theta}\\ \ddot{\theta}\end{bmatrix}=\begin{bmatrix}0 & 1 & 0 & 0\\ 0 & \frac{-(J+md^2)b}{J(M+m)+Mmd^2} & \frac{m^2gd^2}{J(M+m)+Mmd^2} & 0\\ 0 & 0 & 0 & 1\\ 0 & \frac{-mdb}{J(M+m)+Mmd^2} & \frac{mgd(M+m)}{J(M+m)+Mmd^2} & 0\end{bmatrix}\begin{bmatrix}x\\ \dot{x}\\ \theta\\ \dot{\theta}\end{bmatrix}+\begin{bmatrix}0\\ \frac{J+md^2}{J(M+m)+Mmd^2}\\ 0\\ \frac{md}{J(M+m)+Mmd^2}\end{bmatrix}u$$

$$y=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\end{bmatrix}\begin{bmatrix}x\\ \dot{x}\\ \theta\\ \dot{\theta}\end{bmatrix}+\begin{bmatrix}0\\ 0\end{bmatrix}u$$

FIGURA 25 - Matrizes do Espaço de Estado adaptadas para a cadeira
FONTE: Dados da pesquisa.

Sendo que:

As matrizes A são os coeficientes das variáveis de estado, B os coeficientes da ação de controle para os estados presentes; C e D, as saídas presentes em função das variáveis de estado (deslocamento e ângulo).

As matrizes no espaço de estado levam em consideração o peso devido à massa **M** da Cadeira de Rodas Robótica e o peso devido à massa **m** do usuário, que resultam na massa **MCG** do centro de gravidade do sistema, com a sua distância **d** em relação ao referencial inercial.

O modelo matemático da cadeira ocupa a posição do bloco f(u) nessa estrutura, tanto para a aceleração linear como para a angular, e pode ser visto na FIG. 26. Neste mesmo diagrama, apresenta-se a geração do atrito que participa das equações, assim como a perturbação de origem interna, devido à movimentação do usuário que modifica a posição do CG e do valor de **d**, a distância deste à referência inercial.

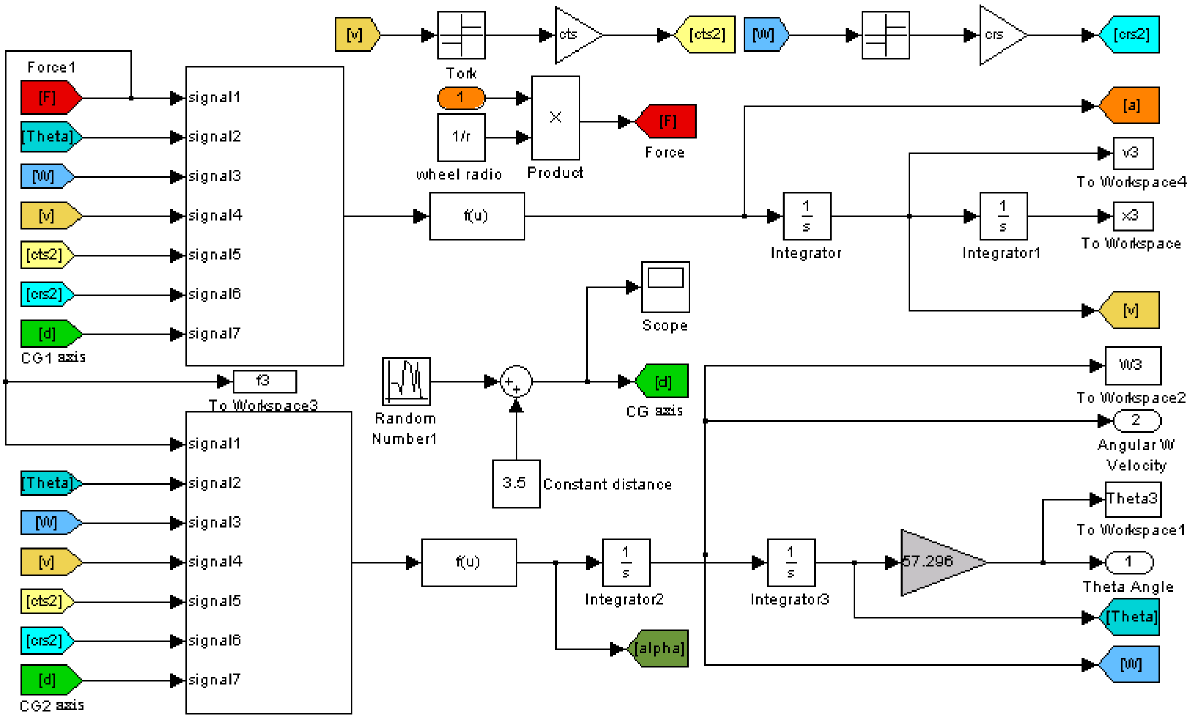


FIGURA 26 - Modelo matemático para equilibrar a cadeira com uma perturbação interna
FONTE: Dados da pesquisa.

5 METODOLOGIA

Neste capítulo, proposta uma metodologia para o desenvolvimento de projetos e controle de cadeira de rodas servo-assistidas, em geral, abrangendo não apenas a parte mecânica e algumas montagens que foram realizadas, como também os controladores e as interfaces conforme a FIG. 27 .

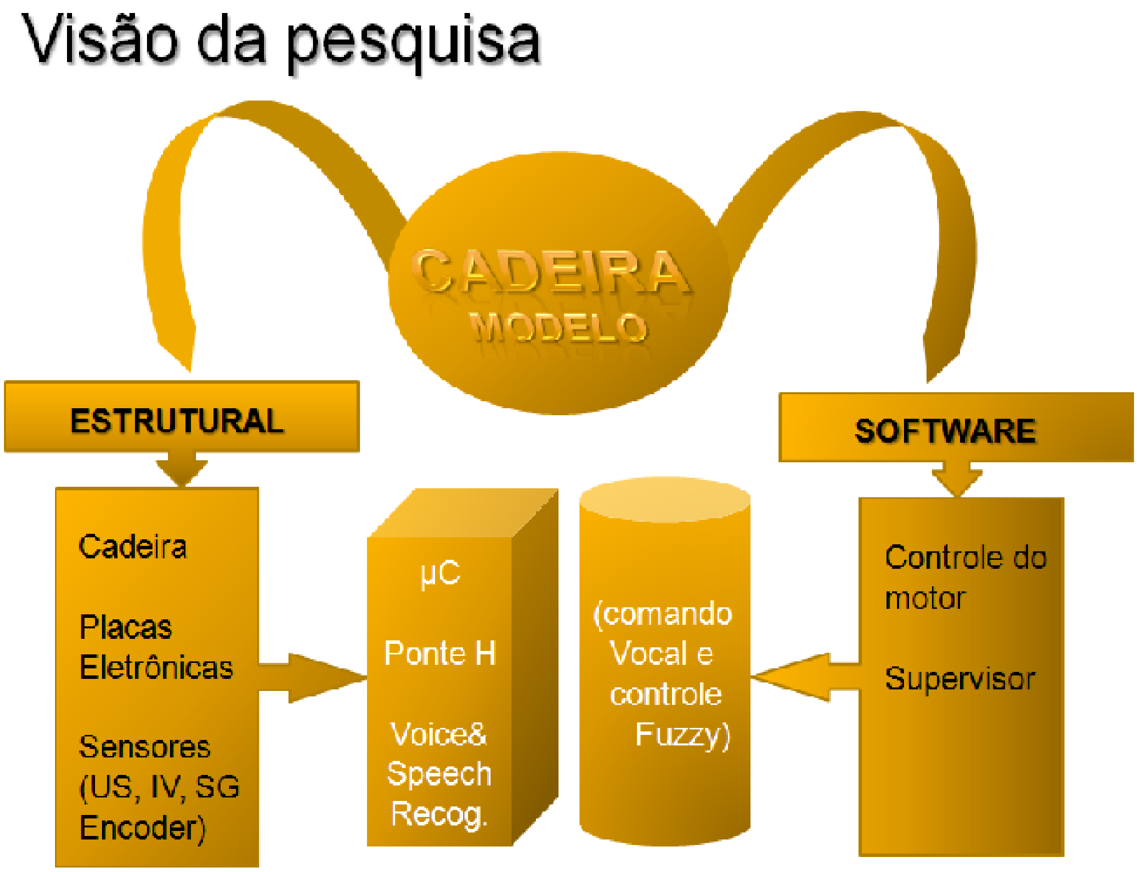


FIGURA 27 - Diagrama em blocos para visualisar o caminho da pesquisa
FONTE: Dados da pesquisa.

Inicialmente, é apresentado um panorama sobre a forma de concepção da cadeira com as suas partes constituintes, de forma a fazer uma ligação daquilo que é simulado com o dispositivo fixo. Posteriormente, são colocadas as metodologias empregadas não só para o reconhecimento de voz por redes neurais artificiais como também para o projeto e sintonia do principal controlador desse dispositivo baseado na Lógica “Fuzzy”.

O desenvolvimento prático da cadeira resultou em um protótipo construído em chassis de alumínio com duas rodas dianteiras livres do tipo denominado “Steers” ou “Castor” e duas rodas diferenciais traseiras, acionadas cada uma por motoredutores DC espelhados de

12 Volts e por uma placa dipH01®, um "driver" da Digital International Projects, cada uma controlada por uma placa dipC02® da Digital International Projects baseada em um microcontrolador PIC®, um netbook cuja placa mãe é compatível PC®, mas com interfaces via USB e sensores diversos, conforme a FIG. 28.

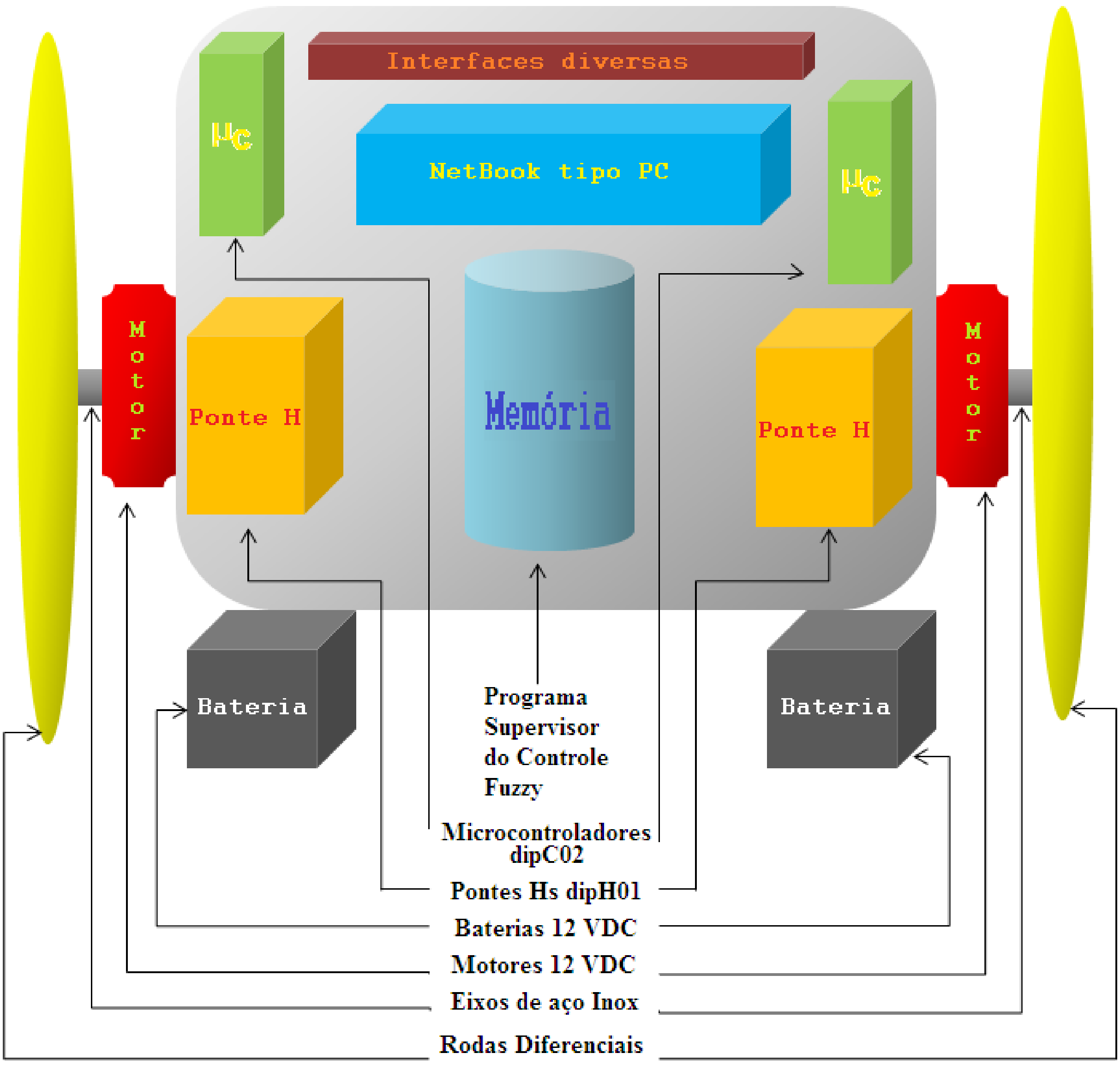


FIGURA 28 - Diagrama em Blocos da cadeira
FONTE: Dados da pesquisa.

Sensoriamento é a palavra chave inclusive para o mais simples comportamento de mobilidade. Um dispositivo que pretende ser automatizado deve ser capaz de sensoriar e processar este sensoriamento de forma a tomar decisões. Os sensores que estão mais associados à mobilidade são aqueles que medem a distancia, as mudanças inerciais e os que medem a estrutura externa ao seu redor. Naturalmente, por trás desse sensoriamento, está um algoritmo (programa) que processa convenientemente essa informação.

5.1.1 O sensor de comando vocal

O principal objetivo deste trabalho foi desenvolver um circuito que permitisse o comando pelo usuário da Cadeira de rodas através da voz. Esta é a entrada para que o sistema como um todo inicie a ação e os movimentos.

A implementação desse sensor foi possível devido à utilização do "Kit Voice Direct" ™, cujo funcionamento será descrito mais adiante.

Não foi considerado, neste trabalho, o comando manual, mas algumas modificações, a fim de permitir que o comando da cadeira de rodas, que feito por meio da voz, possa ser feito com um joystick.

Juntamente com o comando por voz foi desenvolvido outro que possibilitou o controle da cadeira de rodas via microcomputador compatível PC®, ou seja, agora a cadeira de rodas poderá ser comandada pelo micro e voz cujos sinais são enviados à cadeira de rodas por transmissão via rádio ("Wireless") ou fios.

Detalhes desse desenvolvimento estão no corpo da Tese e nos Apêndices, junto à fundamentação teórica.

Utilizando as saídas binárias geradas pelo "VoiceDirect"™, foi desenvolvido um programa em C++ que, introduzido na memória do "netbook" possa interpretar e combinar os sinais com o dos outros sensores, para gerar o sinal de controle pela lógica nebulosa. Isso é feito de modo que seja enviado um código via interface USB para a entrada em cada um dos microcontroladores PIC® que acionam a roda esquerda e direita, que, posteriormente, tratam tais sinais gerando o PWM, que por sua vez enviam, para os respectivos "drivers" da cadeira de rodas, as ordens recebidas de forma vocal.

O sistema de recepção do sinal é composto basicamente por um rádio receptor e entra na interface USB do "netbook". Este trata os sinais recebidos e os transmite para os demais controladores PIC® da cadeira de rodas.

5.1.2 O sensor Encoder

Foram colocados em cada uma das rodas "Encoders" do tipo Bourns® incrementais cujos sinais de saída são um trem de pulsos e cuja resolução está associada à espessura das linhas, à quadratura (direção do giro), e à posição dos sensores (em 90^0). Existe um ponto inicial para estes.

5.1.3 O sensor Célula de Carga "Strain Gage"

Existem diversas aplicações para esse tipo de dispositivo, por isso, analisa-se a empregabilidade para se determinar o tipo a escolher. Células de carga são constituídas de um ou mais sensores extensiômetro de resistência elétrica ("Strain Gage"). Nesse relatório experimental colocado em apêndice, realizaram-se medições de massas com o auxílio de uma célula de carga do tipo cilíndrico, na qual foram feitas medições a fim de demonstrar sua funcionalidade e seus componentes, bem como fazer uma análise dos resultados.

Nesse experimento, realizou-se a construção experimental de uma célula de carga do tipo cilíndrico, que utiliza dois sensores extensiométricos, na qual foram montados os componentes de acionamento mecânicos e uma parte do circuito eletrônico necessário. O principal objetivo desse desenvolvimento não se aplica ao controle "fuzzy", já que se busca demonstrar sua aplicação prática em controles convencionais, realizar medidas experimentais e extrair uma análise sobre estas. O objetivo deste trabalho foi proporcionar o aprendizado sobre o funcionamento desse sensor e sobre seus componentes para uma futura adaptação de um controle moderno por espaço de estados, no qual o peso do usuário da cadeira de rodas e o seu CG são importantes.

No projeto que se apresenta, houve a construção de uma célula de carga em alumínio (Al) do tipo cilindrica, com dois extensiômetros e uma meia ponte de "Wheatstone", que tem a finalidade de medir força, peso, ou pressão, como mostrado na FIG. 29.



(a) (b)

FIGURA 29 - Sensor extensiométrico (a) Projetado no Laboratório e (b) Comercial
FONTE: HBM® do Brasil.

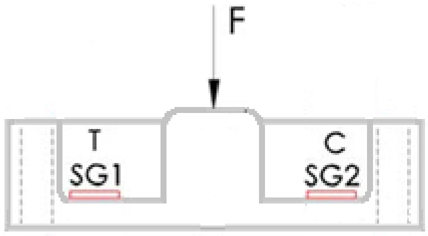


FIGURA 30 - Esquema de funcionamento do “strain gauge” no desbalanceamento da ponte
FONTE: Dados da pesquisa.

Pode-se observar na FIG. 30 que esta célula é autocompensada, pois, quando aplicada uma força, o extensiômetro 1 mede contraindo e o 2, descontraindo pela forma de suas montagens.

Nos apêndices há uma visão mais fundamentada da teoria destes com as suas referências bibliográficas.

Essas células de carga (compostas cada uma por dois “Strains Gages” em ponte de “Wheatstone”) são os sensores responsáveis pelas indicações das reações. Foi usado um total de quatro células de carga denominadas de S_1 a S_4, de acordo com a FIG. 31.

De fato, uma esfera e mola simulam o ser humano e sua movimentação aleatória na cadeira.

Para o cálculo das coordenadas do CG humano, foi utilizado o equilíbrio das forças fs_1, fs_2, fs_3, fs_4 no plano XY, considerando os sensores S_1, S_2, S_3, S_4 aos pares, como indicado na FIG. 31:

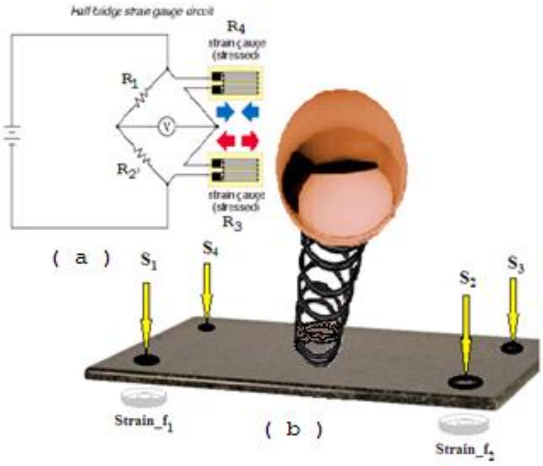


FIGURA 30 - (a) Células de Carga com sensores “Strain Gages” e (b) Sistema Esfera-Mola
FONTE: Dados da pesquisa.

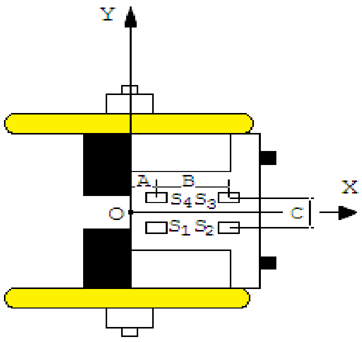


FIGURA 31 - Vista do plano XY com os sensores
FONTE: Dados da pesquisa.

Sendo que:

fs_1, fs_2, fs_3, fs_4 serão as forças nos sensores;
A, B, C, são as distâncias tendo O como referência;

Por meio desse teste, mostrou-se que é possível observar as características de sensores extensiométricos em células de carga e que eles possuem uma larga aplicação, como, por exemplo, sensoriar os movimentos de um usuário de cadeira de rodas, além de variarem conforme a necessidade. Exibiram-se os componentes necessários para o funcionamento dessa célula de carga. Houve a comprovação da variação proporcional de tensão em função da deformação da célula e se observou o comportamento praticamente linear de sua curva. Em função da adequada fixação da célula de carga na bancada de teste através da adição de uma chapa de aço parafusada na célula, obtiveram-se valores satisfatórios nas medições.

Observa-se nos resultados uma pequena variação de tensão em função das massas utilizadas e se pode concluir que, nesse experimento, é possível obter precisão melhor que 1Kg. Limitações podem ser atribuídas ao amplificador utilizado.

É importante ressaltar que o laboratório utilizado não possuía as condições adequadas para realizar as medições, pois o amplificador para meia ponte e os extensiômetros de 120Ω utilizados não eram adequados aos extensiômetros e ponte utilizados. Isso poderia gerar um desbalanceamento inadequado nos resistores e, consequentemente, um desvio na saída de medição ou, até mesmo, perda de sensibilidade. Convêm observar aqui, a extrema importância das técnicas de medidas experimentais onde se prova que a confiabilidade e as incertezas com ocorrências probabilísticas são críticas.

5.1.4 Controladores

Placas controladoras são partes do "hardware" de computadores ou circuitos microprocessados/microcontrolados que comandam outras partes internas ou externas da máquina. Um microcontrolador é um circuito integrado composto por microprocessador programável, memória e interfaces.

5.1.5 Placa supervisora

A placa supervisora é um compatível PC® ("NetBook" da Dell®) adequado para ser o controlador de todo o sistema da cadeira. Como só possui interfaces USB, uma placa conversora sob a forma de uma interface binária foi necessária para a compatibilização com a placa controladora das rodas.

5.1.6 Placa dipC02^{TM4} do Controlador das rodas diferenciais da cadeira

Paralelo aos sinais gerados pelo "netbook", podem ser enviados pelos sensores sinais de comando que entram diretamente na placa controladora do PIC para modificações necessárias nas velocidades e direções através do "driver". Para isso, foi desenvolvido um circuito capaz de enviar tanto os comandos gerados por voz que passam pelo supervisor quanto aqueles gerados pelos sensores externos.

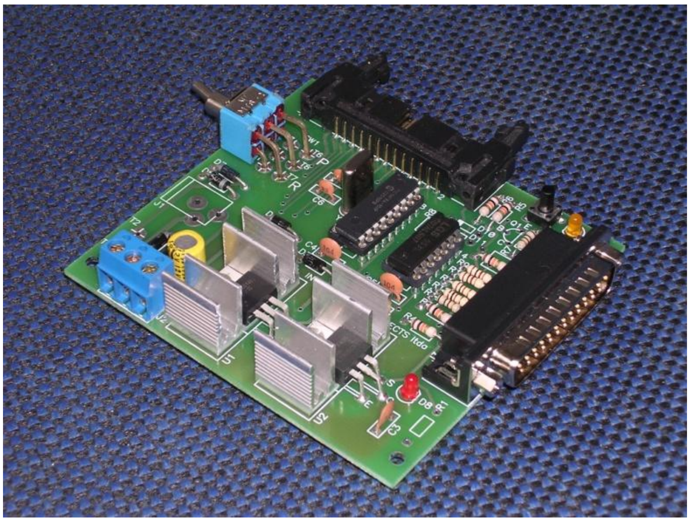


FIGURA 32 - "Lay-out" da placa controladora dos motores fabricada pela Digital International Projects® baseada no PIC16F84®
FONTE: Dados da pesquisa.

A placa controladora dipC02® da Digital International Projects®, é uma unidade eletrônica de automação para trabalhos diversificados.

O microcontrolador utilizado em tal placa é o PIC16F84A® fabricado pela Microchip® (USA), que pode trabalhar em frequências que variam entre 4,10 e 20 MHz.

5.2 Motor: a eletromecânica de acionamento

Motores D.C. são motores que giram sob Corrente Direta de uma bateria ou de uma fonte D.C. para esse tipo de projeto. Eles existem em muitos tamanhos e tipos, mas a função básica de todos eles é a mesma, servem para converter energia elétrica em energia mecânica.

Quando a tensão aplicada através dos controladores por uma bateria ou fonte D.C. se conecta entre as escovas elétricas do motor de D.C., este converte energia elétrica em trabalho mecânico girando um eixo central ligado ao centro das rodas diferenciais da cadeira.

O motor elétrico é o mais conveniente de todas as fontes de energia motriz. É limpo e silencioso, parte imediatamente e pode ser construído grande o bastante para acionar os trens de alta velocidade do mundo ou pequeno o bastante para trabalhar em um relógio de pulso.

5.2.1 Características do motor

O motor DC (com escova) da Mabuschi® é utilizado no projeto que se apresenta com uma redutora para diminuir a sua velocidade e aumentar o Torque normalmente necessário para a Cadeira, tornando-se muito potente para o seu volume. A sua utilização é fácil, pois envolve dois fios apenas (polaridade $\pm$); seu torque é proporcional à corrente; e o seu estado estacionário é constante.

A sua velocidade com a carga é proporcional à voltagem, mas, para trabalhos mais pesados e de baixa velocidade, requer uma redutora, conforme a figura abaixo:

No servo controle requer realimentação (ou "feedback"), de acordo com as equações abaixo:

$$V = iR + L\,(di/dt) + k_e\omega \qquad \text{(eq.5.1)}$$

$$\tau = k_t\, i \qquad \text{(eq.5.2)}$$

Sendo que: V = Voltagem
i = Corrente
R = Resistência
L = Indutância
ω = Velocidade Angular
τ = Torque
k_e = Constante elétrica
k_t = Constante de torque

Nesse projeto, foram utilizados dois motores com redutoras espelhadas, conforme foi dito, para as rodas diferenciais, como em um robô móvel, conforme a FIG. 34:



FIGURA 33 - Conjunto motriz da roda esquerda e direita
FONTE: Dados da internet.

Motores DC requerem drivers para o seu acionamento como uma ponte H, por exemplo.

5.2.2 Ponte H ("Driver"): a eletrônica de acionamento

Os fabricantes de interfaces PWM (Modulação por Largura de Pulso) concorrem para que seus controladores para motores elétricos sejam os mais eficazes. Assim optou-se pela placa dipH02® da Digital International Projects, que atendeu perfeitamente não só este projeto como também a outros que incorporaram produtos da *dip* – Digital International Projects como base, lembrando que há vários modos de se fazer PWM que foram tentados e rejeitados, por uma ou outra razão qualquer. Os apêndices deste capítulo dão

uma boa ideia dos princípios envolvidos e do que fazer, como também do que não fazer. Esta seção é, na realidade, uma pequena contribuição em controle de motor, visto que este dispositivo foi o mais prático para os acionamentos de robôs na oficina de robótica do Laboratório LRSS da UFMG.

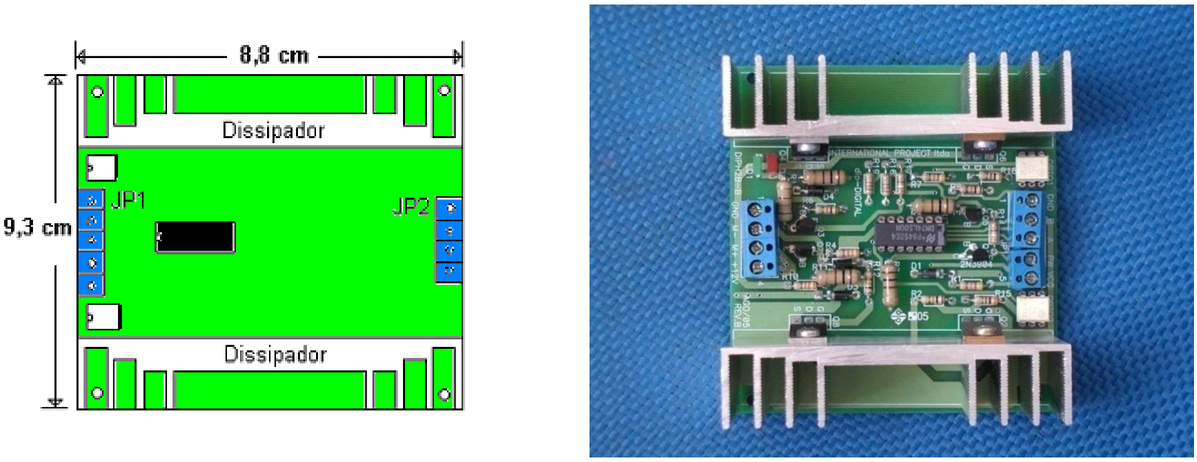


FIGURA 34 - “Lay-out” e a placa do “driver” do motor dipH02® da Digital International Projects®
FONTE: Dados da pesquisa.

Um controlador de motor comercial é mais que simplesmente um circuito para alterar a velocidade do motor. Se a fonte for chaveada (liga-desliga), de tempo em tempo e rapidamente, o motor correrá de algum modo da velocidade entre zero e velocidade máxima. Isso é exatamente o que um controlador PWM faz, ligar o motor em uma série de pulsos. Controlar a velocidade desse motor DC é variar (modular) a largura dos pulsos, consequentemente tem-se a Modulação por Largura de Pulso, conforme a FIG. 36:

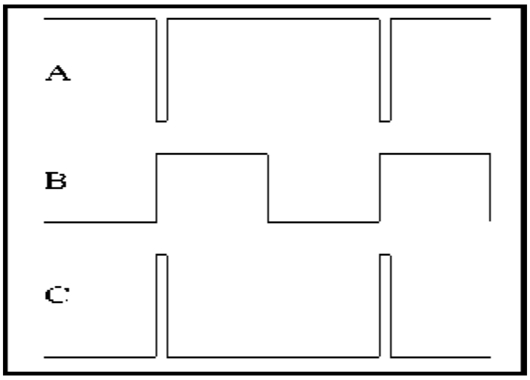


FIGURA 35 - Modulação por largura de pulsos
FONTE: Dados da pesquisa.

Controladores de motor de ponte H também usam PWM. Projetar um circuito de ponte H de PWM que seja seguro e "à prova de usuário" é bastante difícil, mas existem circuitos confiáveis no mercado, como o que foi utilizado neste estudo.

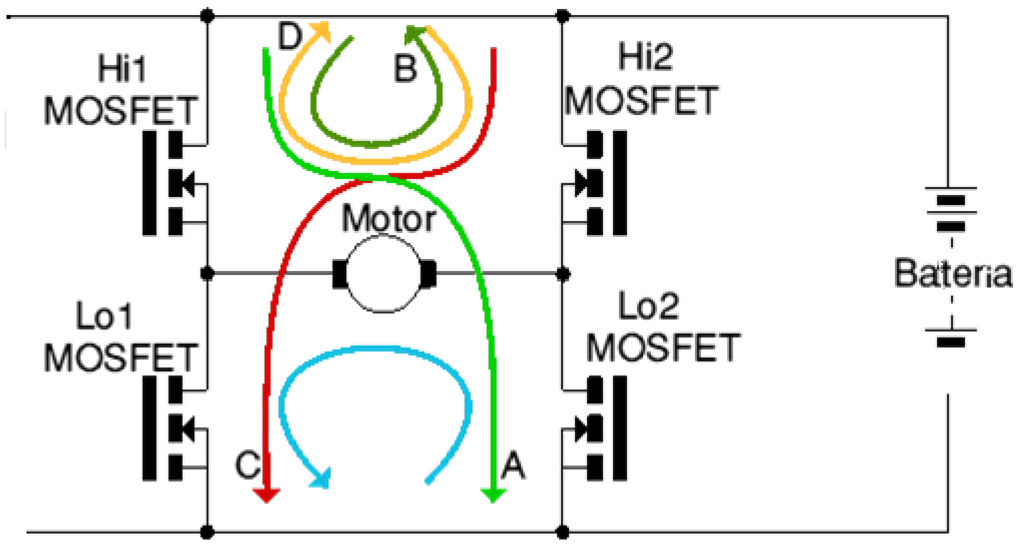


FIGURA 36 - Ponte H para controle PWM de um motor DC
FONTE: Dados da pesquisa.

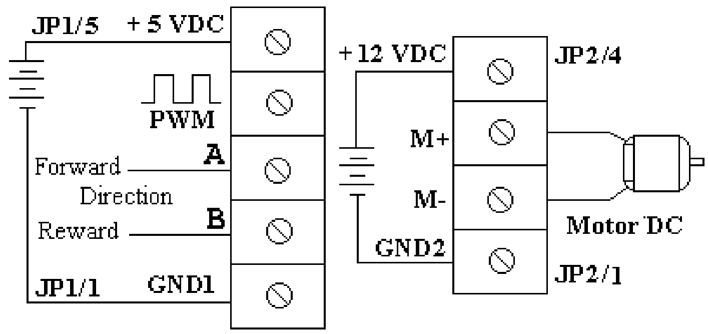


FIGURA 37 - Esquema de ligação dos motores para Modulação por Largura de Pulsos (do inglês PWM)
FONTE: Dados da pesquisa.

5.3 O conjunto estrutural da cadeira de rodas

Inicialmente, como foi dito, a cadeira foi projetada e simulada para atender a determinados comportamentos que elevassem os graus de liberdade alcançados por um deficiente com as cadeiras convencionais. Nesse sentido, o desenvolvimento se apoiou em um modelo matemático que contemplasse esses movimentos, no plano e fora dele.

A cadeira está baseada em uma plataforma de alumínio que possui um movimento de balanço em torno do eixo de duas rodas traseiras pneumáticas que participam do

acionamento de forma independente, ou seja, são de tração diferencial. Isso permite o conjunto de movimentos estabelecidos, tais como: girar à direita, girar à esquerda, inclinar, avançar e recuar, entre outros.

A roda frontal gira livre, sem estar vinculada a nenhum acionamento, servindo apenas para o apoio da plataforma no solo, sem prejudicar o giro em solo.

O projeto original foi orientado para que a situação do equilíbrio fosse provocada por uma partida com um elevado torque e o CG estivesse acima do referencial.



FIGURA 38 - O autor na montagem da cadeira de rodas robótica e equilibrista no laboratório LRSS da UFMG
FONTE: Dados da pesquisa.

Parâmetros físicos da cadeira

TABELA 3 - Parâmetros físicos da cadeira de rodas equilibrista

Parâmetro	Descrição	Valor
Mad	Massa do conjunto da roda direita	895 g
Mae	Massa do conjunto da roda esquerda	895 g
Maf	Massa do conjunto da roda frontal	280 g
Mmd	Massa do motor impulsor direito	1565 g
Mme	Massa do motor impulsor esquerdo	1565 g
Mrd	Massa do redutor direito	330 g
Mre	Massa do redutor esquerdo	330 g
Mcc	Massa do chassis da cadeira	1790 g
Mbd	Massa da bateria direita	2580 g
Mbe	Massa da bateria esquerda	2580 g
MCG	Massa do Centro de Gravidade	
l	Largura do chassis da cadeira	0,30 m
c	Comprimento do chassis da cadeira	0,40 m
Cad	Coordenada do conjunto da roda direita	0 x 0 x 210mm
Cae	Coordenada do conjunto da roda esquerda	0 x 0 x -210mm
Caf	Coordenada do conjunto da roda frontal	300 x –20 x 0mm

Cmd	Coordenada do motor impulsor direito	-30 x 117 x 110mm(*)
Cme	Coordenada do motor impulsor esquerdo	-80 x 100 x -110mm(*)
Crd	Coordenada do redutor direito	0 x 0 x -110mm
Cre	Coordenada do redutor esquerdo	0 x 0 x -110mm
Ccc	Coordenada do chassis da cadeira	160 x 0 x 0mm
Cbd	Coordenada da bateria direita	5 x 50 x 32mm
Cbe	Coordenada da bateria esquerda	5 x 50 x -32mm(*)
CCG	Coordenada do Centro de Gravidade	11,3817 x 41,6940 x 0
g	Aceleração da gravidade	9,81 m/s^2
ccs	Coeficiente de atrito seco do eixo do motor DC	0,0005
ccv	Coeficiente de atrito viscoso do eixo do motor DC	0,0005

Variáveis	**Descrição**
xG	Posição linear X do Centro de Gravidade da cadeira
dxG/dt ou v_x	Velocidade linear X da cadeira
d^2xG/dt^2 ou dv_x/dt ou a_x	Aceleração linear X da cadeira
yG	Posição linear Y do Centro de Gravidade da cadeira
dyG/dt ou v_y	Velocidade linear Y da cadeira
d^2yG/dt^2 ou dv_y/dt ou a_y	Aceleração linear Y da cadeira
zG	Posição linear Z do Centro de Gravidade da cadeira
dzG/dt ou v_z	Velocidade linear Z da cadeira
d^2z/dt^2 ou dv_z/dt ou a_z	Aceleração linear Z da cadeira
θ	Posição angular theta da cadeira no plano xy
$d\theta/dt$ ou w_z	Velocidade angular theta da cadeira no plano xy
$d^2\theta/dt^2$ ou dw_z/dt ou α_z	*Aceleração angular theta da cadeira no plano xy*
ϕ	Posição angular phi da cadeira no plano zx
$d\phi/dt$ ou w_y	Velocidade angular phi da cadeira no plano zx
$d^2\phi/dt^2$ ou dw_y/dt ou α_y	Aceleração angular phi da cadeira no plano zx
φ	Posição angular psi da cadeira no plano zy
$d\varphi/dt$ ou w_x	Velocidade angular psi da cadeira no plano zy
$d^2\varphi/dt^2$ ou dw_x/dt ou α_x	Aceleração angular psi da cadeira no plano zy

FONTE: Dados da pesquisa.
Obs.: Todas as coordenadas no sentido XYZ foram tomadas em relação à referência inercial no meio do eixo entre as duas rodas traseiras e diferenciais. Como existe perfeita simetria na estrutura da cadeira, não foi necessário modificar a posição de nenhuma peça para compensar uma diferença das coordenadas devido ao estudo do CG nas seções anteriores.

5.4 Reconhecimento de comandos vocais por redes neurais

O sensoriamento de sons e principalmente de comandos vocais é muito utilizado em navegação robótica para o desvio de obstáculos e deslocamentos em ambientes não estruturados. Contudo, uma tarefa altamente desejável, porém de difícil implementação, é a detecção e reconhecimento de sons e comandos vocais confiáveis, devido às fortes

interferências que o sistema de sensoriamento de sons sofre do meio ambiente, como variações de temperatura e correntes de ar, além de ruídos. Além disso, a forma como o sinal de som varia é intrinsecamente não linear. A aplicação de redes neurais para o reconhecimento de comandos vocais vem se mostrando muito eficiente em diversas pesquisas realizadas. A utilização de redes neurais normalmente é indicada quando o processo envolvido é não linear e/ou difícil de modelar matematicamente, sendo exatamente este o caso em questão.

Transdutores para o reconhecimento de voz são muito utilizados em robótica, dada a sua boa relação custo-benefício na medição de sons para a detecção de comandos vocais que evitem obstáculos, para a correção dos erros cumulativos do sistema de odometria, para o controle, para a construção de mapas do ambiente de navegação e para a orientação em ambientes desestruturados (KOMIYA, 2000). O espectro do som pode ser visto na FIG. 39.

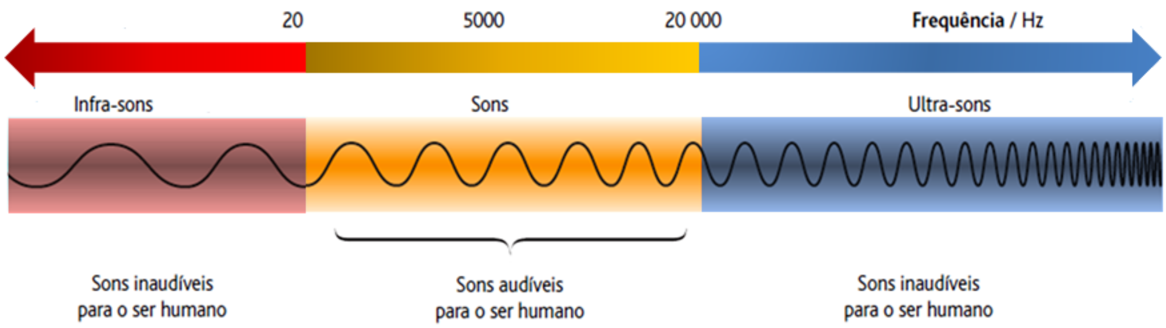


FIGURA 40 - Espectro da frequência do Som
FONTE: Dados da pesquisa.

Tanto a construção de mapas e a orientação da cadeira de rodas como a de controle por reconhecimento de voz, normalmente, baseiam-se em vocábulos simples, tais como parar, avançar, recuar e suas derivações relativas. Contudo, uma melhoria substancial na realização de tais tarefas seria obtida pela incorporação de outros vocábulos tipicamente encontrados no ambiente de navegação, como virar (direita e esquerda) e inclinar. O reconhecimento de vocábulos por reconhecimento de voz, tais como os citados é o ponto central deste artigo, tendo como aplicações-alvo as cadeiras de rodas, tal como a robótica (FIG. 1), ainda em desenvolvimento. Como trabalhos relacionados, foram referenciados em Pires e Nunes (2002), baseando-se em lógica nebulosa, e Simpson e Levines (2002), que utilizaram redes neurais no reconhecimento de voz para a diferenciação comandos.

5.4.1 O reconhecimento de voz em robótica

Em geral, além do sensoriamento interno (odometria), as cadeiras de rodas robóticas necessitam de sensoriamento exteroceptivos para sua navegação confiável. Além de "encoders" óticos (transdutores responsáveis pela odometria) acoplados as duas rodas diferenciais de tração, a cadeira de rodas robótica deve possuir um sistema de sensoriamento e reconhecimento de voz, além de um sistema de sensoriamento de obstáculos por ultrassom.

As cadeiras de rodas com as características citadas possuem, em geral, um sistema de controle híbrido, ou seja, um sistema de controle orientado por comportamentos cujas condutas de mais alto nível baseiam-se em uma abordagem de controle que tanto pode ser convencional (moderno, robusto e outros) como não convencional (baseado em inteligência artificial) (GASÓS; ALMEIDA, 1999). Um exemplo de um comportamento desse tipo seria um que fosse responsável por mapear o ambiente de operação da cadeira de rodas.

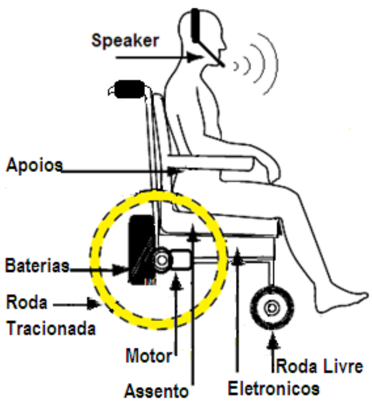


FIGURA 39 - A cadeira de rodas robótica com usuário no comando vocal
FONTE: Dados da pesquisa.

Os dados necessários à sobrevivência da cadeira de rodas em seu ambiente de operação devem ser fornecidos pelos "encoders" das rodas e, principalmente, pelo sistema de sensoriamento por ultrassons, ao passo que o sistema de sensoriamento de reconhecimento de voz fica responsável por fornecer dados para os comportamentos de mais alto nível. O presente trabalho busca obter maior proveito do sensoriamento de reconhecimento de voz além de suas funções básicas, extraindo informações que possam ser importantes para as decisões tomadas pelos níveis superiores.

A utilização de reconhecimento de voz para o sensoriamento externo da cadeira de rodas móvel deve-se ao fato de que estes sistemas apresentam excelente relação custo benefício, sendo muito baratos, além de fornecerem um conjunto de informações suficientes

para permitir que ums cadeira de rodas móvel possa se deslocar com relativa segurança em ambientes parcialmente estruturados. Na cadeira de rodas, utiliza-se uma interface baseada em redes neurais de fabricação da Sensory Inc® na FIG. 41 que pode ser treinada para aprender os comandos necessários ao controle e navegação. Dessa forma, resolve-se o problema de levar tal processamento ao sistema supervisor com o inevitável aumento de atraso devido ao programa.

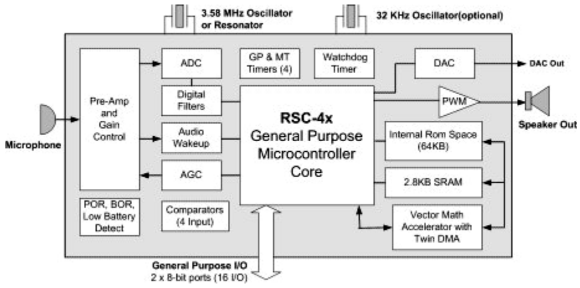


FIGURA 40 - Interface da Sensory Inc.® completa para captura do Som
FONTE: Sensory Inc.

Essa interface da família RSC-4x® se baseia em processadores projetados para trazer recursos de áudio avançados em produtos de consumo sensíveis e embutidos. Baseado em um microcontrolador de 8 bits, a família RSC-4x® integra blocos de processamento digital e analógico otimizado em uma solução de chip único capaz de reconhecimento de voz preciso, de alta qualidade e baixa taxa de dados de fala comprimida. Os produtos podem usar um ou todos os recursos em uma única aplicação. Além do melhor desempenho no reconhecimento, a família RSC-4x® proporciona maior integração "on-chip" de recursos, incluindo um pré-amplificador de microfone, unidades "twin" (DMA), vetor acelerador, multiplicador de "hardware", três temporizadores, e até 4,8 Kbytes de RAM.

Um sistema completo pode ser construído com outros componentes adicionais mínimos como: bateria, alto-falante, microfone e alguns resistores e capacitores.

Em aplicações que exigem menos tempo de discurso ou conjuntos de comandos menores, a RSC- 464® oferece 64 Kbytes de memória ROM e é recomendada para redução de custos.

No entanto, muitas vezes a simples informação de que existe um obstáculo a certa distância é insuficiente para que a cadeira de rodas possa se deslocar com segurança.

Frequentemente, é necessário identificar, no ambiente de operação da cadeira de rodas, alguma referência geométrica que pode ser utilizada para que a cadeira de rodas se

localize no ambiente, encontre um posto de recarga ou simplesmente corrija o erro cumulativo da odometria.

O reconhecimento de referências vocais utilizando reconhecimento de voz é uma tarefa complexa, quer pela sensibilidade do reconhecimento de voz a distúrbios ambientais, principalmente variações de temperatura e correntes de ar, quer pela não linearidade do processo, de difícil modelagem matemática. Por esses motivos, as redes neurais tornam-se principais candidatas para utilização. Pode-se constatar ao final deste artigo que o seu emprego permitiu a obtenção de excelentes resultados.

Neste trabalho, foram utilizadas duas variantes do algoritmo “backpropagation”: a primeira, denominada “Resilient Back-Propagation (Rprop)” (RIEDMILLER, 1994), por sua eficácia na convergência; e a segunda, baseada na associação da otimização de Levenberg-Marquardt (FINSCHI, 1996) com a otimização por regularização Bayesiana. A otimização pelo método Levemberg-Marquardt possibilita uma convergência robusta e em um número menor de iterações, embora requeira substancialmente mais tempo de processamento e recurso de memória; enquanto a regularização Bayesiana evita o aprendizado excessivo, impedindo que neurônios além do número mínimo necessário aprendam e assim se especializem (decorem) nos exemplares do conjunto de treinamento. Através da informação fornecida sobre o número efetivo de neurônios envolvidos no aprendizado, possibilita-se a tarefa de dimensionamento da topologia da rede.

5.4.2 Referências vocais

O conjunto de referências vocais que se deseja reconhecer foi escolhido de acordo com os comandos de operação da cadeira de rodas que já está parcialmente desenvolvida para operar em ambientes como residências e escritórios, além de ruas e calçadas. Como pode ser observado a seguir, nas imagens dessas ondas, foi utilizada uma frase em inglês que ilustra bem uma sentença com alguns tipos de palavras.

O numero de comandos para o controle é muito maior, e a formação do banco de dados dessas ondas vocais demanda um tempo maior para que várias amostras sejam armazenadas do que aquele de que se dispunha no desenvolvimento deste estudo conforme a FIG. 42 a seguir:



FIGURA 41 – Formação do Banco de Dados de comandos vocais
FONTE: Dados da pesquisa.

Dessa forma, procurou-se escolher seis (6) comandos frequentemente encontrados para tais ambientes; as referências (classes) selecionadas foram:

a) avançar;
b) direita;
c) esquerda;
d) inclinar;
e) parar;
f) recuar.

5.4.3 Aquisição dos dados

Existem trabalhos realizados com o mesmo propósito que este, porém utilizando lógica nebulosa (INDELICATO, 1997). Neste trabalho, a aquisição dos dados foi feita pela geração analógica da envoltória do reconhecimento de voz, conversão da envoltória para digital e determinação de algumas das características consideradas mais representativas, tais como o tempo de duração do sinal, tempos de subida e descida, tempos e valores de pico, áreas e outras.

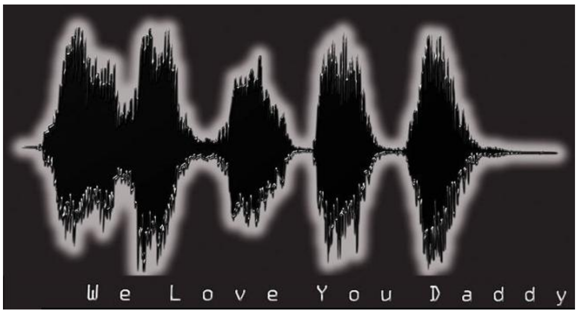


FIGURA 42 - Exemplo de ondas vocais em uma frase
FONTE: Dados da pesquisa.

Na comparação, seguiu-se o mesmo procedimento para a determinação das características de entrada da rede neural, e uma descrição detalhada sobre a obtenção dessas características pode ser encontrada nos trabalhos deste autor (RENNO *et al.*, 2000).

Para isso, foi selecionada a onda vocal da palavra "Daddy" da frase original.

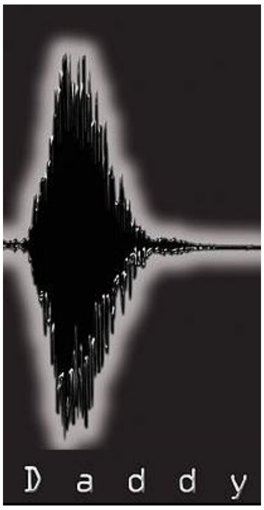


FIGURA 43 - Onda vocal selecionada
FONTE: Dados da pesquisa.

Eliminando os valores negativos, determinou-se a sua envoltória da parte positiva ou a envoltória superior isolada. Esse passo é fundamental para identificação de uma onda vocal no processo de reconhecimento de voz em projetos de robótica e serve como preparação dos passos que serão realizados a seguir.

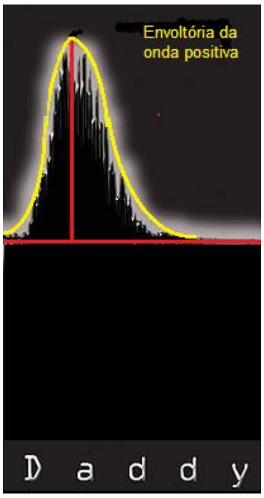


FIGURA 44 - Seleção de pontos em uma das ondas vocais selecionadas
FONTE: Dados da pesquisa.

Foram adquiridos conjuntos de nove características do sinal da onda vocal. Estas características são extraídas da envoltória FIG. 46:

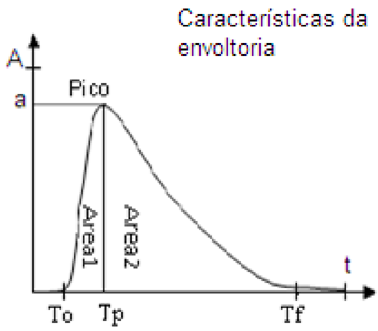


FIGURA 45 - Características da envoltória
FONTE: Dados da pesquisa.

T_a = Tempo de subida (T_p-T_0).
T_d = Tempo de descida (T_f-T_p).
T_0 = Tempo de início da onda.
T_p = Tempo do pico da onda.
Tf = Tempo do final de onda.
Área = Área total sob a envoltória
Área1 = Área sob a envoltória em T_a.
Área2 = Área sob a envoltória em T_d.
a = Amplitude máxima do sinal.

Para a eficiência de cada classe (referência), devem ser utilizados vários sinais de ondas vocais para diversos timbres de voz.

5.4.4 Reconhecimento de referências vocais utilizando redes neurais

O desenvolvimento deste trabalho se deu no ambiente Matlab® da empresa americana Mathworks®, em plataforma PC®. Uma vez definido o tipo de rede, sua topologia e conjunto de pesos sinápticos, pode-se, então, efetuar a implementação em processador embutido, a bordo da cadeira de rodas robótica e equilibrista.

Antes do treinamento neural, foi realizado o pré-processamento estatístico, denominado Análise dos Componentes Principais (PCA), a fim de detectar características redundantes, bem como reduzir a dimensão da entrada da rede. No caso específico deste trabalho, a análise PCA reduziu a dimensão dos vetores de dados de entrada de 9 para apenas 3, retendo 99% da variância total, ou seja, desprezando as características cujo somatório das variâncias no novo sistema não excedia 1%.

Após os testes com vários dimensionamentos de rede (número de camadas internas e número de neurônios por camada interna) e, principalmente com a ajuda da regularização Bayesiana, chegou-se à seguinte configuração apropriada: entrada de dimensão 3 (encontrada pelo pré-processamento PCA), 16 neurônios na única camada escondida e 5 neurônios na camada de saída, competitivos do tipo “winner takes all” (Haykin,1999), um para cada classe.

Além dos conjuntos de treinamento e validação, foi utilizado um terceiro, para teste definitivo do poder de generalização. Os três conjuntos eram mutuamente exclusivos, ou seja, sem qualquer sobreposição, e continham situações de todas as classes e distâncias na faixa considerada.

O treinamento da rede neural, conforme procedimentos realizados, utilizou a função TRAIN denominada RNAVOZ (Matlab®), no Apêndice 5, que treina esta rede neural (rede neural artificial para o reconhecimento de voz).

5.4.5 Resultados obtidos

A tabela a seguir resume o percentual de acerto associado a cada uma dos 6 tipos de referências vocais de comando da cadeira de rodas, obtido por cada uma das redes implementadas. Os valores indicados referem-se somente ao conjunto de teste, não envolvido no treinamento.

TABELA 4 - Acertos pela RNA para cada comando

Tipo deReferência Geométrica	RPROP	Levenberg-Marquardt
Avançar	94,3%	97,7%
Direita	100,0%	100,0%
Esquerda	100,0%	100,0%
Inclinar	87,4%	91,5%
Parar	93,4%	100,0%
Recuar	100,0%	96,7%
Geral	95,8%	97,6%

FONTE: Dados da pesquisa.

5.5 O controle Fuzzy ou controle baseado em conhecimento

Existem diversos dispositivos projetados pelo ser humano que apresentam elevadas características não lineares, inerentemente instáveis e variáveis no tempo. Tais dispositivos são difíceis de serem modelados e controlados. Os métodos clássicos que são usados para sistemas lineares apresentam limitações quando aplicados a sistemas com essas características. Existem abordagens alternativas modernas para estudar, modelar e controlar tais sistemas, mas algumas são denominadas de não convencionais, mais conhecidas como baseadas em conhecimento e incluem o uso de ferramentas de inteligência artificial (SHAW; SIMÕES, 1999).

Este trabalho descreve o desenvolvimento da modelagem e controle de um sistema com essas características citadas, baseado em um controle não convencional, este com o emprego da lógica “fuzzy”. Esta se situa em um patamar que permite a um leigo (técnicos

ortopédicos, médicos bio-engenheiros e outros), em relação ao controle convencional, projetar e executar um projeto, desde que tenha conhecimento do seu funcionamento. Este trabalho investiga essas novas técnicas, aplicadas a um caso real de uma cadeira de rodas para desabilitados físicos tetraplégicos, com a opção da inclinação.

5.5.1 Controle "Fuzzy"

Definiu-se que, na teoria dos conjuntos "fuzzy" (ou nebulosos), a suposição de que os elementos chave do pensamento humano não são informações que possuam uma classificação exata, mas um conjunto de classes de objetos que são conjuntos com fronteiras difusas (de onde vem o seu nome), ou seja, a transição de classes não é tão contínua como nos controladores, modernos e robustos nem abrupta como na lógica digital (CLAUSER; MCCONVILLE; YOUNG, 1969; HAUSER; SANTOS, 2002).

O conhecimento do comportamento de uma planta pode ser expresso de forma mais natural usando uma combinação de variáveis linguísticas tais como: Ângulo (elevado ou pequeno), Velocidade (elevada ou pequena), todos relevantes para a formação de uma lógica de controle (RENNO; BASTOS FILHO, 2000).

A geração de uma tabela que estabeleça uma correlação entre variáveis de estado de uma planta, como a cadeira de rodas, e a ação necessária para controlá–la, de certa forma, é uma *modelagem nebulosa baseada em conhecimento*.

TABELA 5 - Regras simplificadas para os quadrantes do controle "fuzzy" da cadeira de rodas

Ang Vel	$\theta<90^0$	$\theta=0$	$\theta>90^0$
$\omega<0$	$F>0$ ω contra		$F<0$ ω a favor
$\omega=0$		$F=0$	
$\omega>0$	$F>0$ ω a favor		$F<0$ ω contra

FONTE: Dados da pesquisa.

Sendo que: θ = Angulo de inclinação;

ω = Velocidade angular;

F = Força de atuação ou ação *U*.

Considera-se o ângulo e a velocidade iguais a zero. Nesse caso, os motores não tem o que fazer, ou seja, não fazem nada, pois sua ação é nula.

Considera-se outro caso: a velocidade é pouco positiva. Assim, há que se mover a plataforma na mesma direção e com uma velocidade baixa para compensar.

Como poderá ser visto a seguir, na implementação do controle "fuzzy", houve necessidade de dividir a tabela anterior do controlador em duas regiões, o que gerou uma adequação das funções de pertinência, de um controlador Grosso e um Fino (em torno do ângulo de equilíbrio de 90 Graus), de forma a melhorar a sensibilidade.

As adaptações necessárias surgiram em função da introdução do atrito nas equações originais do modelo matemático da cadeira no capítulo 5, tornando o sistema mais realista para posterior simulação e aplicação.

Os resultados obtidos serviram para validar os modelos matemáticos reduzidos e posteriormente os controladores.

Os controladores "fuzzy" são a mais importante aplicação da teoria da Lógica Fuzzy na área de controle inteligente. O trabalho é diferente dos controladores convencionais devido ao conhecimento que é usado ao invés do desenvolvimento das equações diferenciais que descrevem o sistema.

Uma característica importante dos controladores nebulosos ("fuzzy") é a habilidade em executar controles multiobjetivos, mesmo em situações conflitantes, de forma a se obter um bom compromisso na estratégia de controle, embora as estratégias de controle "fuzzy" nasçam da experiência e de experimentos do projetista ao adquirir conhecimento intuitivo sobre o processo em detrimento de modelos matemáticos complexos comprovados.

Portanto, a implementação do controle nebuloso ("fuzzy") nasce sob uma forma linguística assemelhada à humana, com suas evasões.

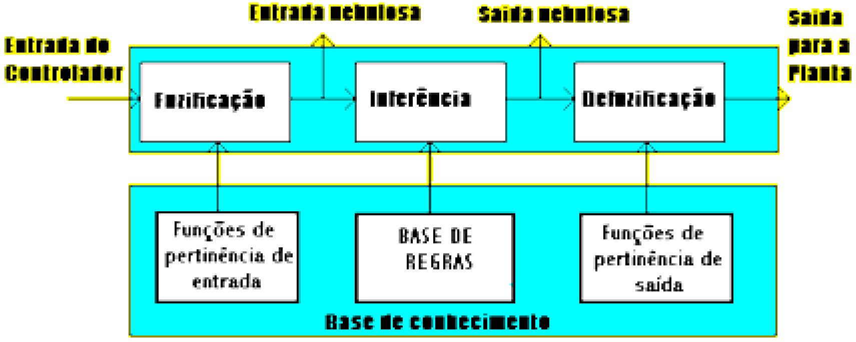


FIGURA 46 - Diagrama em blocos de um controlador "Fuzzy"
FONTE: Dados da pesquisa.

Técnicas usando lógica nebulosa foram utilizadas para controle do pêndulo invertido, que, de certa forma, constituiu a fase preliminar para o estudo deste trabalho. Mas estas técnicas apenas controlavam a posição do pêndulo, sem levar em consideração outras condições dinâmicas existentes, como foi realizado neste trabalho para a cadeira equilibrista que possui um Centro de Gravidade (CG) ligado virtualmente a um referencial local.

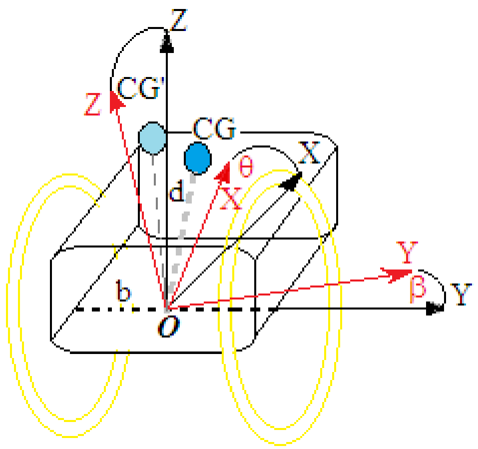


FIGURA 47 - Diagrama espacial da cadeira robótica e os seus graus de liberdade
FONTE: Dados da pesquisa.

Esse conhecimento pode ser expresso de uma forma mais natural usando uma combinação das variáveis linguísticas relevantes para formação de uma lógica de controle conforme se vê a seguir:

SE variável de entrada X1 = x_1 **E** variável de entrada X2 = x_2 **Então** a saída Y= y

5.5.2 Aplicações da Lógica "Fuzzy"

Define-se agora quando o Controle com a Lógica "Fuzzy" é aplicável em termos gerais. O emprego de Controle "Fuzzy" é recomendável:

a) para processos muito complexos;
b) para processos altamente não lineares;
c) se o processamento de conhecimento especializado (formulados linguisticamente) deve ser feito quando não há modelo matemático conhecido.

O emprego do Controle "Fuzzy" não é recomendável se:

a) a teoria convencional de controle dá resultados satisfatórios;
b) um modelo matemático adequado e de fácil aplicação já existe;
c) outra combinação híbrida é mais eficiente;
d) o problema não é solucionável dentre os padrões aceitáveis.

Existem vários exemplos de como a Lógica “Fuzzy” é aplicada na realidade e que são compatíveis com este trabalho:

a) controle simplificado de robôs;	[Fuji Eletric, Toshiba]
b) controle eficiente e estável;	[Nissan]
c) combinação de Lógica “Fuzzy” e Redes Neurais.	[Matsushita]

5.5.3 Controle de uma Cadeira de Rodas Robótica e Equilibrista no plano XZ

O problema se resume a controlar uma cadeira de rodas com movimentos para frente, para trás, parar e equilibrar uma plataforma móvel em torno de seu CG (Centro de gravidade) que pode se mover apenas em dois sentidos esquerda e direita.

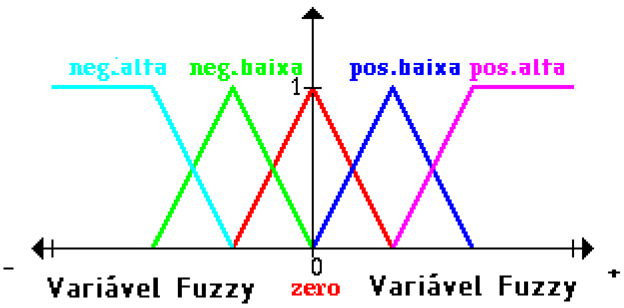


FIGURA 50- Gráfico com as funções de pertinência de um controlador “fuzzy”
FONTE: Dados da pesquisa.

Primeiramente, define-se o que é uma variável de saída negativa, positiva, entre outros referenciais da plataforma da forma:

a) muito negativo (azul claro);
b) pouco negativo (verde);
c) zero (vermelho);
d) pouco positivo (azul);
e) muito positivo (rosa).

O mesmo é feito para o ângulo entre a plataforma e o solo e a velocidade angular deste ângulo. Definem-se, a partir de agora, várias regras que dizem o que fazer em certas situações, levando ao processo de defuzzificação:

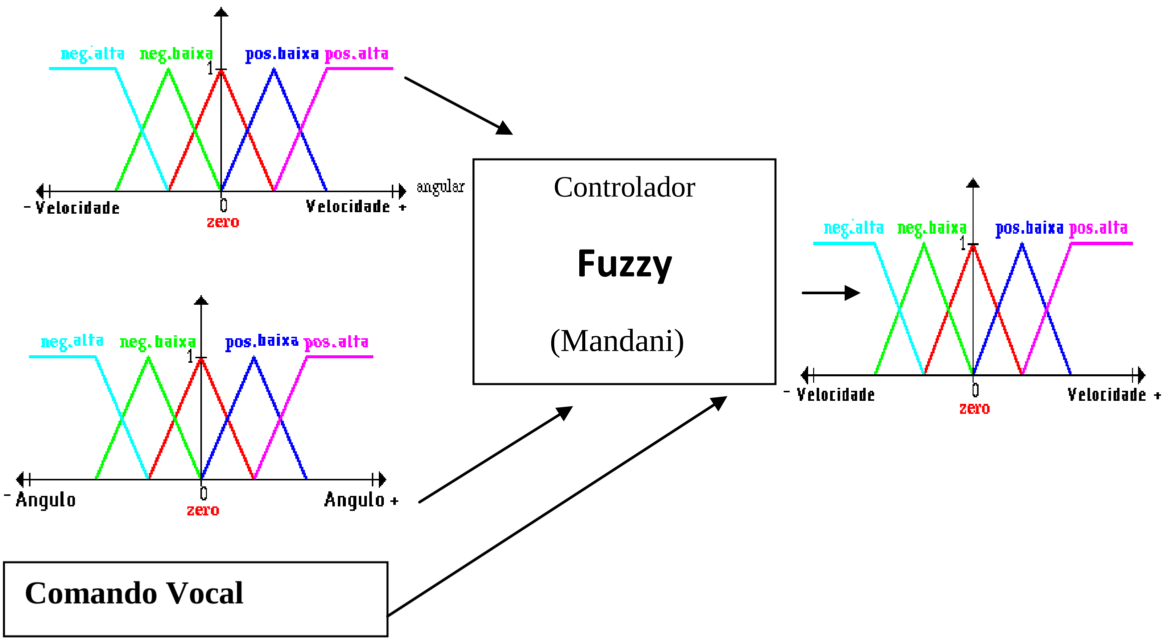


FIGURA 51 - Diagrama em blocos com entradas e saídas de um controlador “fuzzy”
FONTE: Dados da pesquisa.

Evolução de regras:

a) SE o comando é inclinar e o ângulo é zero e a velocidade angular é zero, ENTÃO a velocidade total é zero;
b) SE o comando é inclinar e o ângulo é zero e a velocidade angular é negativa baixa, ENTÃO a velocidade total será negativa baixa;
c) SE o comando é inclinar e o ângulo é zero e a velocidade angular é positiva baixa, ENTÃO a velocidade total é pouco positiva;
d) ...

Define-se agora o comando desejado e dois valores explícitos para o ângulo e a velocidade angular para calcular a velocidade total. Considera-se a situação atual:

a) um comando para o movimento da cadeira (o foco desse trabalho foi Inclinar);
b) um valor atual para o ângulo;

c) um valor atual para a velocidade angular.

Aplicando a regra:

a) SE o comando é inclinar e o ângulo é zero e a velocidade angular é zero, ENTÃO a velocidade total é zero (para os valores selecionados na situação anterior);
b) SE o comando é inclinar e o ângulo é zero e a velocidade angular é negativa baixa, ENTÃO a velocidade total será negativa baixa;
c) SE o comando é inclinar e o ângulo é positivo baixo e a velocidade angular é zero, ENTÃO a velocidade total será positiva baixa;
d) SE o comando é inclinar e o ângulo é positivo baixo e a velocidade angular é negativa baixa, ENTÃO a velocidade total será zero;
e) SE o comando é inclinar e o ângulo é zero e a velocidade angular é zero, ENTÃO a velocidade total é zero;
f) SE o comando é inclinar e o ângulo é zero e a velocidade angular é negativa baixa, ENTÃO a velocidade total será negativa baixa;
g) SE o comando é inclinar e o ângulo é zero e a velocidade angular é positiva baixa, ENTÃO a velocidade total é pouco positiva. E assim por diante.

5.5.4 Defuzzificação

Quando, finalmente se tem o cálculo do grau de pertinência dos conjuntos nebulosos das variáveis de saída (uma consequência das regras definidas anteriormente), através do processo de inferência, é necessário aplicar o processo de defuzzificação, cuja finalidade é a de converter as ações de controle nebulosas em ações de controle não nebulosas (ou “Crisp”). Existem vários métodos, média dos máximos, centro de gravidade, singleton e outros, sendo que se utilizou neste trabalho o do Centro de Gravidade (CoG) ou da centroide, como se vê a seguir:

$$u = \frac{\sum_{i=1}^{n} (A_i . \mu_i) . Centróide_i}{\sum_{i=1}^{n} A_i} \qquad \text{(eq.5.3)}$$

Sendo que:

u = É a ação de controle de nossa saída deffuzificada;
A_i = É uma área dos n conjuntos i nebulosos da saída;
μ_i = Grau de pertinência da $Centróide_i$;
$Centróide_i$ = É COG ou CG com o grau de pertinência correspondente.

Nossa saída defuzzificada segue a combinação dada na base de regras como no exemplo a seguir:

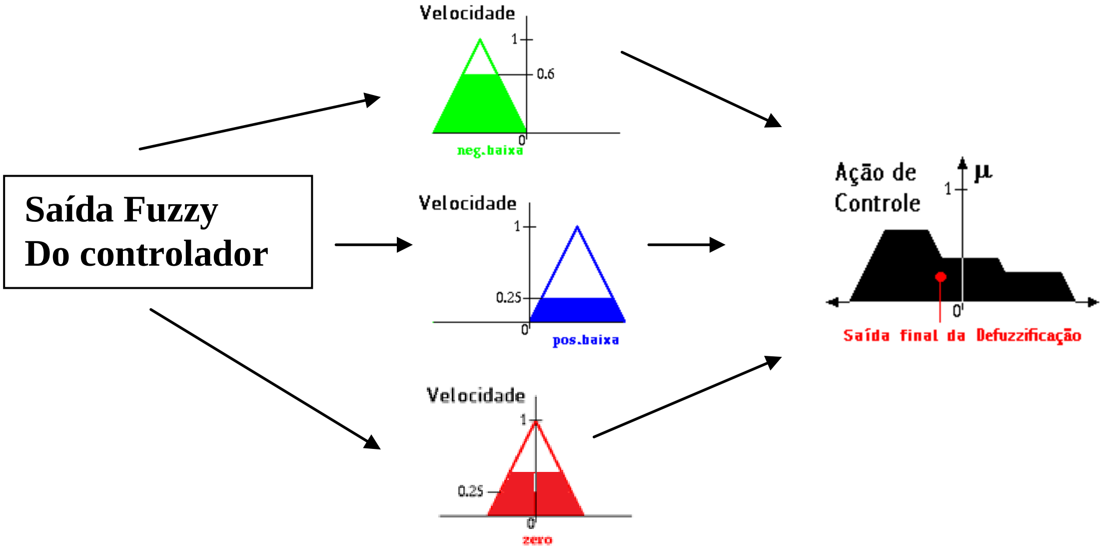


FIGURA 52 - Diagrama em blocos com as e saídas de um controlador “fuzzy”Gráfico com as funções de pertinência de um controlador “fuzzy”
FONTE: Dados da pesquisa.

Foi utilizada a ferramenta Simulink® do MatLab® para simulação do controle “fuzzy”, de forma preliminar, com o comando vocal a incluir o modelo matemático da cadeira segundo o método de Newton, a fim de testar a realimentação das variáveis de estado (Ângulo e Velocidade Angular) na entrada e a Ação de controle na saída (Força) do controlador “fuzzy”, obtendo-se o diagrama da FIG. 52:

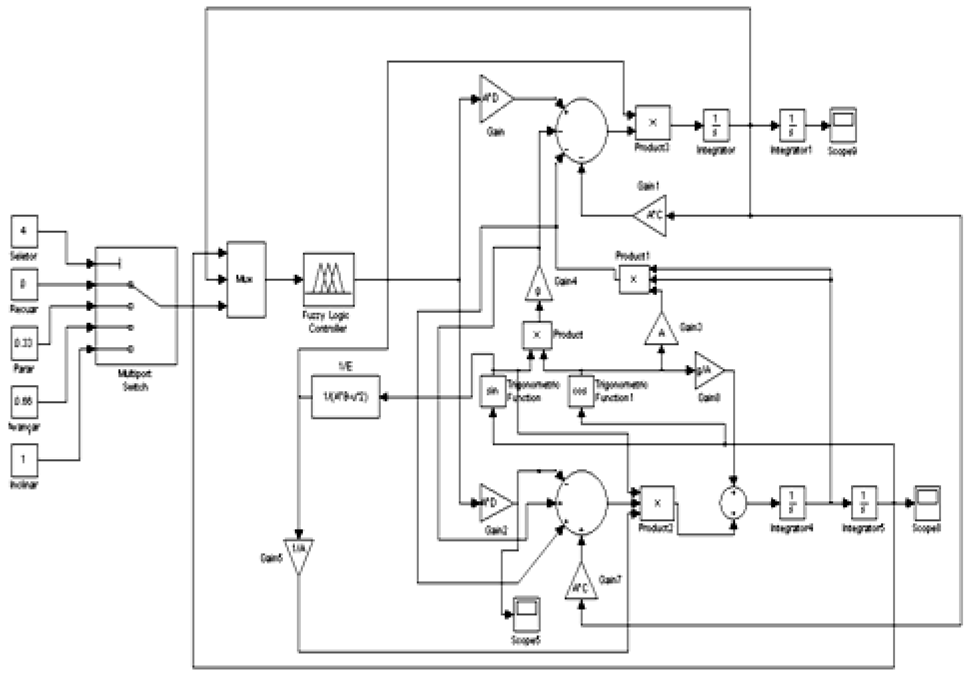


FIGURA 53 - Diagrama da planta no Simulink® para o controlador “fuzzy” com o comando vocal
FONTE: Dados da pesquisa.

As simplificações e melhorias no modelo dinâmico da cadeira com parâmetros reais levaram a um controle que, apesar de simulado, aproxima-se bem mais de um caso real com suas equações mais precisas, conforme demonstram os gráficos obtidos adiante nesta simulação.

Atualmente, os controladores “fuzzy” provaram ser uma das mais importantes aplicações da lógica “fuzzy”. No controle convencional, determinístico, é necessário o desenvolvimento de funções matemáticas bem complexas, que serão responsáveis pela estabilidade da planta a ser controlada. Já no controle fuzzy, aproximativo, não é necessário este conhecimento matemático da planta em questão, pois, neste caso, o sucesso das estratégias de controle nasce da experiência e de experimentos do projetista ao adquirir conhecimento intuitivo sobre o processo, em detrimento daqueles modelos matemáticos complexos e já comprovados na literatura (RENNO; LIMAII; PINTO, 2012).

Várias técnicas que empregaram a lógica “fuzzy” foram bem divulgadas na literatura para o controle do pêndulo invertido (ABRANTES, 2008), que, de certa forma, foi uma fase preliminar para o estudo deste trabalho, mas que se resume apenas a uma barra inclinada e apoiada que sofre uma impulsão em sua base. A cadeira equilibrista, em seus vários estágios de equilíbrio, apresentada de forma estilizada na FIG. 53 assemelha-se ao pêndulo invertido em alguns aspectos, embora tenha as suas próprias características dinâmicas. Essa cadeira possui um centro de gravidade (cgw), assim como o usuário (cgh),

resultando em um Centro de Gravidade total (CG) ligado virtualmente a um referencial inercial local O, mas que se modifica com o movimento do usuário.

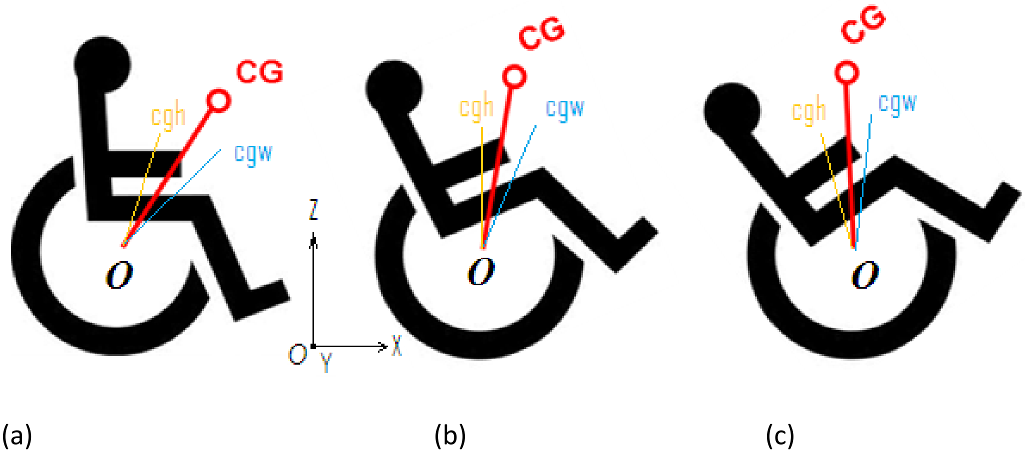


FIGURA 54- A cadeira equilibrista como um pendulo no início instável (a), em transição (b) e no equilíbrio (c)
FONTE: Dados da pesquisa.

Verificada a validade do modelo, partiu-se para a concepção do controlador "fuzzy". Este será o responsável por gerar o valor aproximativo que, através dos acionadores, gerará o torque T aplicado às rodas da cadeira de raio r, com o intuito de manter o CG com um ângulo de inclinação (θ) de + 90º.

Inicialmente as variáveis fuzzy das entradas do controlador foram criadas conforme a Fig. 53; são elas, o ângulo Theta (θ) e a velocidade angular Omega (ω), assim como foi criada a variável saída do mesmo que, sob a forma de um valor correspondente à tensão V_a (a ser amplificada pelos "drivers"), será aplicada aos motores e que através daquele torque T será aplicado às rodas de raio r cujo resultado é a força de atuação F.

Uma vez definidas as variáveis linguísticas de entrada e saída do controlador, construiu-se o que se chama de base de regras do controlador (CLAUSER; MCCONVILLE; YOUNG, 1969), vista na TAB. 6, em que a primeira letra de cada função de pertinência indica se o valor da variável é negativo (N) ou positivo (P), o que significa igualmente qual será o sentido de giro das rodas ("Forward" ou "Reward"); e a segunda letra indica se o valor é pequeno (S), médio (M), grande (H) ou enorme (B), correspondendo às diversas potências aplicadas, o que resulta na velocidade de giro das rodas e suas respectivas velocidades. Adicionalmente, tem-se siglas para ZE (em torno de 0°) e EQ (em torno de $+90^o$).

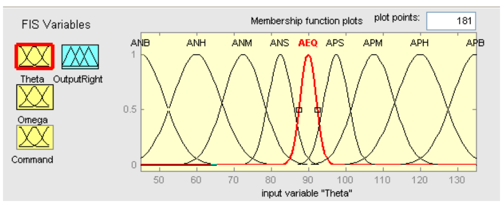
Nomenclaturas na lógica fuzzy são definidas livremente pelos seus projetistas, na forma de mnemônicos que induzem a um valor, grandeza, etc.

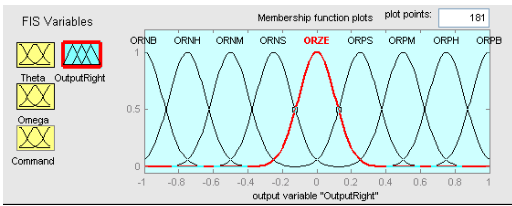
TABELA 6 - Base de regras do controlador “fuzzy”

ω \ θ	NB	NH	NM	NS	EQ	PS	PM	PH	PE
NH	PB	PB	PH	PH	PM	NM	NM	NH	NH
NM	PE	PH	PH	PM	PP	NM	NM	NM	NH
NS	PH	PH	PM	PM	ZE	NS	NS	NM	NM
ZE	PH	PM	PM	PS	**ZE**	NP	NM	NH	NH
PS	PM	PM	PS	PS	ZE	NM	NM	NH	NH
PM	PH	PM	PM	PM	NS	NM	NH	NH	NB
PH	PH	PH	PM	PM	NM	NH	NH	NB	NB

Fonte: Dados da pesquisa.

A tabela foi construída sob a forma de funções de pertinência que expressam bem o comportamento das variáveis “fuzzy”, Theta, Omega e a saída “Output” da ação para um dos motores, conforme a FIG. 54





(a)
(b)

(c)

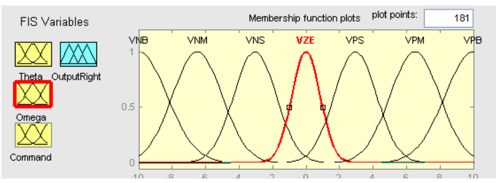


FIGURA 55 - Variáveis Fuzzy (a) Ângulo Theta (b) Velocidade Angular ω a saída (c) do controlador Fuzzy e suas funções de pertinência
FONTE: Dados da pesquisa.

A opção por funções de pertinência “normais” ou “triangulares”, embora possa levar a resultados semelhantes a escolha de uma ou outra, significa uma ação de controle mais suave ou menos, ou seja, ríspida e mais incômoda para o usuário. Particularmente, preferimos as sigmoides (normais) por apresentarem um comportamento monotônico e ganho mais natural, semelhante ao que observamos em processos físicos onde suas naturezas são tipicamente não lineares.

Nos apêndices deste capítulo, descrevem-se as fundamentações teóricas da lógica “fuzzy”, de redes neurais biológicas, de modo a estabelecer uma analogia com os modelos de redes neurais artificiais. Na sequência, uma possível definição de rede neural, sua

organização, características e histórico da abordagem baseada nesses modelos são apresentados.

5.6 Projeto convencional de controle por Espaço de Estados

Do ponto de vista do processamento de dados, o projeto de um controle pelo método de Espaço de Estados possui grandes vantagens por realizar todos os seus cálculos através da álgebra matricial e por ser realizado no próprio domínio do tempo. Além disso, a ligação entre a modelagem de sistemas dinâmicos de multicorpos realizada e o projeto de controle pelo Espaço de Estados é automática, pois a forma final das equações encontradas na modelagem, depois de linearizadas, é aquela requerida para o projeto final do controle.

A representação das equações de estado é escrita desta forma:

$$\frac{dx}{dt} = \dot{x} = Ax + Bu \quad e \quad y = Cx + Du \qquad \text{(eq.5.4)}$$

Sendo que:

x = É um vetor nx1 que representa os estados (aqui são ângulos, deslocamentos e as velocidades correspondentes);
u **=** É um escalar representando a entrada (uma força ou torque no caso desta cadeira).
y = É um escalar representando a saida.
A, B, C e D = São matrizes de estado que estabelecem uma relação entre entrada e saída.

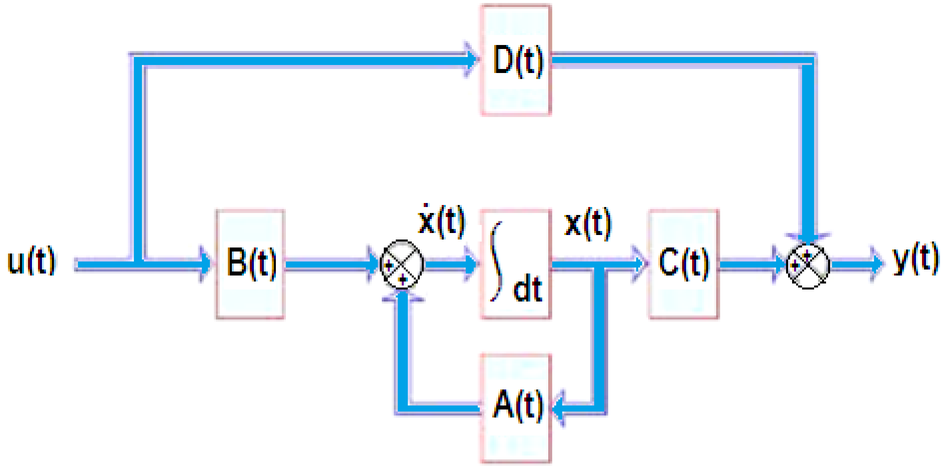


FIGURA 56 - Diagrama em blocos de um modelo por espaço de estados
FONTE: Dados da pesquisa.

Para que seja possível trabalhar nessa linha, além de se ter domínio da controlabilidade e da observabilidade, há que ter bem definidos: o estado, as variáveis de estado, o vetor de estado e o Espaço de Estados.

Tal sistema é constituído, conforme visto no modelamento do capitulo 5, por um sistema impulsor de massa **M** e o chassis fixo a este de massa **m** e de comprimento **c**. Mas esse conjunto encontra sua equivalência no **CG** de massa equivalente e distancia **d** à referência inercial ***O*** situada no centro do eixo traseiro. A cadeira pode mover e girar na horizontal (eixo y) e inclinar em relação a este plano, girando sobre o eixo z. Esse sistema, como foi visto na cinemática do capitulo 4, poderia possuir vários graus de liberdade, pois, além de avançar no plano, pode inclinar e girar (Roll) tanto para a esquerda como para a direita. No entanto, por decidir-se restringir alguns movimentos (por ex. Roll), visto que não se teria condições de processamento em tempo hábil, mesmo que se estivesse realizando apenas uma simulação.

Aquelas equações da dinâmica do sistema obtidas no capítulo 4, efetivamente duas equações diferenciais de 2ª Ordem, resultam em quatro equações diferenciais de 1ª Ordem que descrevem igualmente o mesmo comportamento dinâmico do sistema. Esse vetor dimensional (4x1) pode ser descrito como:

$X = [\, x_1, x_2, x_3, x_4]$

Sendo que:

$x_1 = \theta$
$x_2 = \dot{\theta}$
$x_3 = x$
$x_4 = \dot{x}$

Estabelecidas estas Variáveis de Estado, é possível definir suas derivadas imediatas e aquelas baseadas nas equações do movimento do capítulo 4, de forma que:

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = -\frac{\left(\left(\left(\left(\frac{ac}{ab-\cos^2 x_1}\right)x_4 + \left(\frac{g\,\mathrm{sen}\,x_1\cos x_1}{ab-\cos^2 x_1}\right) + \left(\frac{a\,\mathrm{sen}\,x_1}{ab-\cos^2 x_1}\right)x_2^2 + \left(\frac{ad}{ab-\cos^2 x_1}\right)u\right)\cos x_1 + g\,\mathrm{sen}\,x_1\right)\right)}{a}$$

$$\dot{x}_3 = x_4$$

$$\dot{x}_4 = -\left(\frac{ac}{ab-\cos^2 x_1}\right)x_4 + \left(\frac{g\,\mathrm{sen}\,x_1\cos x_1}{ab-\cos^2 x_1}\right) + \left(\frac{a\,\mathrm{sen}\,x_1}{ab-\cos^2 x_1}\right)x_2^2 + \left(\frac{ad}{ab-\cos^2 x_1}\right)u$$

(eq.5.4)

Sendo que:

a, b, c, d = É uma combinação matemática de coeficientes relacionados aos parâmetros físicos da cadeira[3].

Consideremos o vetor de variáveis **X** e as matrizes **A** e **B** de estados onde a equação geral do movimento deste sistema determinado linearizado onde se obtém:

$$X = \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix} \quad B = \frac{1}{J(M+m)+Mmd^2}\begin{bmatrix} 0 \\ J+md^2 \\ 0 \\ md \end{bmatrix} \quad u$$

(eq.5.5)

$$A = \frac{1}{J(M+m)+Mmd^2}\begin{bmatrix} 0 & J(M+m)+Mmd^2 & 0 & 0 \\ 0 & -(J+md^2)b & m^2gd^2 & 0 \\ 0 & 0 & 0 & J(M+m)+Mmd^2 \\ 0 & -mdb & mgd(M+m) & 0 \end{bmatrix}$$

(eq.5.6)

Em primeira observação, nota-se que tal sistema é possivelmente instável, pois a matriz **A** tem dois autovalores nulos e outro positivo, mas é controlável, visto que a matriz de controlabilidade $\mathbf{M_c}$ possui a inversa $\mathbf{M_c^{-1}}$ conforme abaixo:

$$M_c^{-1} = (4M+m)^2\begin{bmatrix} 0 & 0 & \frac{(M+m)}{(4M+m)^2} & \frac{mc}{2(4M+m)^2} \\ \frac{(M+m)c}{(4M+m)} & \frac{mc^2}{2(4M+m)} & 0 & 0 \\ 0 & 0 & -\frac{c^2}{6g} & -\frac{c^3}{9g} \\ \frac{(4M+m)c^3}{6g} & \frac{(4M+m)c^4}{9g} & 0 & 0 \end{bmatrix}$$

(eq.5.7)

[3] Atenção: ao longo deste trabalho, algumas letras podem causar confusão pelas coincidências em locais e definições diferentes sendo interessante recorrer à nomenclatura utilizada.

A partir das equações diferenciais obtidas nos itens anteriores da parte 1 (malha aberta), foi obtido o modelo do sistema em espaço de estados (matrizes A, B, C e D).

Existem dois procedimentos que devem ser executados para obter as matrizes do espaço de estado: o primeiro deles, conforme o item seguir, é utilizar o comando do Matlab TF2SS(GS), fazendo-se a conversão direta da Função de Transferência do Sistema para o Espaço de Estado, e o outro é o desenvolvimento exposto na apresentação anexa, através de uma metodologia segundo a qual se levam em consideração os elementos armazenadores de energia e as equações da dinâmica associadas a estes.

As matrizes A são os coeficientes das variáveis de estado, B, os coeficientes da ação de controle para os estados presentes, C e D, as saídas presentes em função das variáveis de estado (deslocamento e ângulo).

Conforme observado no item 1 desta parte, obteve-se, conforme se mostra abaixo, as matrizes das equações de estado diretamente a partir da função de transferência obtida no item do modelo da cadeira de rodas para a inclinação. Comparando essas matrizes com as matrizes obtidas no caso do exemplo do pêndulo invertido, possuem uma diferença pelas mudanças que ocorrem na cadeira devido à inércia e na massa do usuário.

Calculando as matrizes de controlabilidade, segundo Pc=[B A*B A*A*B A*A*A*B] (CHEN, 2001), obtendo-se o determinante de Pc, com rank=4 e, conforme Q=[C’ A’*C’ A’^{2}*C’] de observabilidade, sendo o rank de Q=3, verifica-se que essas propriedades se aplicam ao sistema controlável e observável, pois não poderia ser zero.

Determinando os pólos apropriados para o sistema em malha fechada que podem ser obtidos da forma vista anteriormente, foi obtido um controlador por observador de estados com o comando K = lqr(A,B,Q,R) do Matlab®, sendo R (de lqr) um peso que minimiza a função de custo utilizando as novas matrizes de estado AC=[(A-B*K)],...,com os valores de K obtemos o diagrama simplificado de controle conforme a FIG. 56:

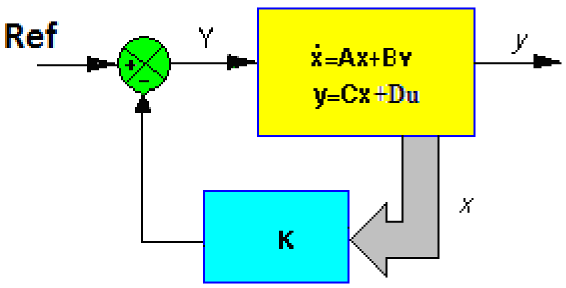


FIGURA 57- Diagrama de um controlador LQR (modernode estados) com realimentação e ganhos K
FONTE: Dados da pesquisa.

Uma função do tipo degrau na referência serve como uma primeira simulação de perturbação neste sistema de controle conforme a FIG. 57.

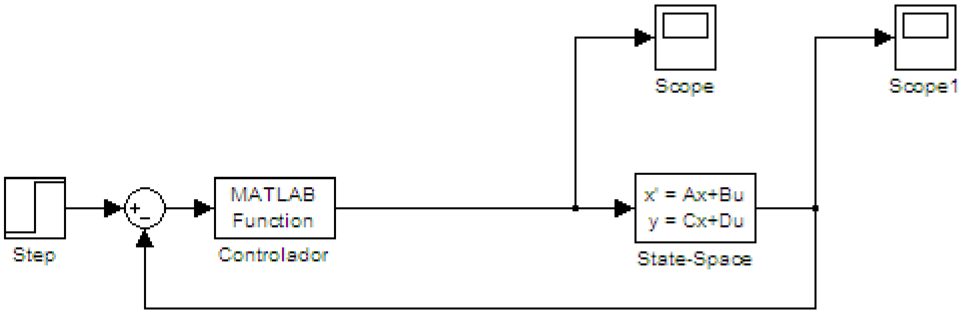


FIGURA 58 - Diagrama do Simulink de um controlador moderno de estados realimentado
FONTE: Dados da pesquisa.

5.6.1 Métodos de determinação do CG

Apresenta-se aqui um estudo que traz certa inovação para o caso do Centro de Gravidade (CG) de uma cadeira de rodas, calculado em tempo real e que o diferencia das técnicas usuais empregadas na estabilização de dispositivos como este.

A proposta desta etapa do trabalho foi a de sensoriar as forças exercidas pelo usuário sobre uma cadeira de rodas equilibrista para possibilitar o cálculo do centro de gravidade resultante. Uma das dificuldades iniciais a ser superada no trabalho foi a da definição do centro de gravidade da pessoa (CG humano), da cadeira (CG da cadeira) e a composição de ambos para formar o centro de gravidade combinado (CG total). Em uma primeira etapa, foi desenvolvido o protótipo da cadeira e realizado os respectivos testes para determinar os seus diversos CGs estáticos. Em uma segunda fase, fez-se o cálculo do centro de massa humano. O objetivo principal foi o cálculo do CG resultante para, em seguida, efetuar os respectivos testes simulados e uma descrição do efeito deste na dinâmica da cadeira.

A escolha do CG resultante ou global do sistema cadeira/pessoa para o desenvolvimento da modelagem matemática implica em uma tarefa nem sempre tão fácil , por diversas causas, entre elas, metodologias mais eficazes para o cálculo de CG humano (cgh). O centro de gravidade do corpo humano depende do tipo de pessoa e da posição relativa de cada parte do corpo. Essa definição implica que o corpo humano seja considerado como composto de vários segmentos rígidos e que, para cada segmento, seja conhecido o respectivo CG. De

um modo geral, considera-se que o CG de jovem adulto em posição ereta normal está localizado na 2° vértebra dorsal. No entanto, em cada movimento do corpo, haverá a variação da posição do CG global do conjunto usuário/cadeira. O movimento, isto é, o deslocamento relativo das partes do corpo rígido, em que o corpo está dividido, provoca alteração constante da resultante do CG humano.

Outros fatores que afetam o CG humano são a idade da pessoa e o sexo. Com a idade, há uma tendência na diminuição da altura percentual do CG. No adulto jovem, o CG encontra-se 15% mais baixo que em uma criança de um ano. Para a mesma idade, altura e peso, verifica-se que o sexo feminino possui o CG localizado entre 1% a 2% mais abaixo que o masculino.

Para se efetuar o cálculo do CG total do conjunto, considera-se a distância de um eixo de referência com a massa de cada parte do corpo e da cadeira. Calculam-se as coordenadas com as equações seguintes:

$$x_{total} = \frac{\sum x_i \times w_i}{\sum w_i} \qquad y_{total} = \frac{\sum y_i \times w_i}{\sum w_i} \qquad z_{total} = \frac{\sum z_i \; w_i}{\sum w_i} \qquad \text{(eq.5.5)}$$

Sendo que:

x_{total} , y_{total} , z_{total}	representam as coordenadas do centro de massa do sistema composto;
x_i , y_i , z_i	representam as coordenadas do centro de massa de cada parte que constitui o corpo e a cadeira;
$\sum w_i$	é a soma dos pesos de todas as partes que constituem o conjunto.

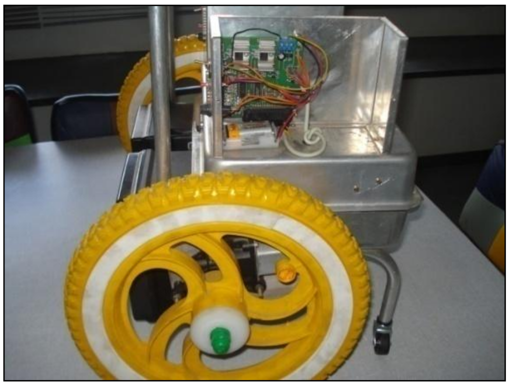


FIGURA 59 - O protótipo da cadeira utilizado para testes
FONTE: Dados da pesquisa.

Para que o mesmo fosse possível, foi necessário construir uma placa no assento, que se inclina de acordo com os movimentos do usuário e, para isso, foram utilizados extensômetros, que são sensores de carga responsáveis pelas indicações das reações. Utilizou-se o total de quatro extensômetros identificados de S_1 a S_4, de acordo com a FIG 61 e 62 adiante. Para validação desse protótipo de cadeira de rodas, foram utilizados, adicionalmente, uma esfera e mola que simulam o corpo humano e sua movimentação aleatória na cadeira.

A escolha do CG (Centro de Gravidade) do sistema cadeira/pessoa para o desenvolvimento do modelamento matemático implica em uma tarefa difícil por citar material para o cálculo de CG humano.

O centro de gravidade (CG) é o termo mais usado popularmente para expressar o centro de massa (CM) de um corpo. O centro CG é simplesmente uma posição média da distribuição da força peso. A definição de centro de massa e centro de gravidade são diferentes: a primeira se refere à distribuição de massa e a outra, à distribuição da força peso. Desde que a força peso exercida em um corpo de massa M é proporcional a sua massa, o CG e o CM se referem ao mesmo ponto. No corpo humano, o centro de gravidade (CG) coincide com o centro de massa (CM).

O centro de gravidade do corpo humano depende da posição relativa de cada parte do corpo. Esta definição implica que o corpo humano seja considerado como composto de vários segmentos rígidos e que, para cada segmento, seja conhecido o respectivo CG. De um modo geral, considera-se que o CG de um jovem adulto em posição ereta normal está localizado na 2° vértebra dorsal. No entanto, em cada movimento do corpo haverá a variação da posição do CG global do corpo. O movimento, isto é, o deslocamento relativo das partes do corpo rígido, em que o corpo está dividido, provoca alteração constante da resultante do CG humano. Ao variar, o CG humano interfere na composição do CG resultante, podendo levar o sistema a uma instabilidade a ser corrigida pelo controlador "fuzzy" (RENNÓ; LIMAII; PINTO, 2013).

Outros fatores que afetam o CG é a idade da pessoa e o sexo. Com a idade, há uma tendência da diminuição da altura percentual do CG. No adulto jovem, o CG encontra-se 15% mais baixo que em uma criança de um ano. Para a mesma idade, altura, peso verifica-se que o sexo feminino possui o CG localizado entre 1% a 2 % mais abaixo que o masculino.

5.6.2 Método de determinação do CG do corpo com base nos CG dos segmentos

É possível calcular as coordenadas do CG a partir de percentuais de cada parte do corpo; estes valores percentuais já foram tabelados para estudos em competições olímpicas. As tabelas contêm as distribuições normalizadas para a massa de cada segmento do corpo. Este modelo considera cada segmento como um corpo rígido e, portanto, cada segmento possui um CG que não se altera durante o deslocamento relativo dos vários segmentos. A alteração das posições relativas dos vários segmentos provoca a alteração do CG global. É considerado que a massa do corpo com 100% e que cada segmento tem uma massa percentual. Este modelo dispensa o conhecimento da massa total da pessoa para efetuar o cálculo. Para determinar a massa aproximada de um segmento de uma determinada pessoa, basta aplicar a tabela.

TABELA 7 - Percentual de massa do segmento

Percentual de massa do segmento (%)		
Segmento	**Homens**	**Mulheres**
Cabeça & Pescoço	8.96 / 6.94	8.20 / 6.68
Tronco	46.84 / 43.46	45.00 / 42.58
Antebraço	3.25 / 2.71	2.90 / 2.55
Braço	1.87 / 1.62	1.57 / 1.38
Mão	0.65 / 0.61	0.50 / 0.56
Perna	10.50 / 14.16	11.75 / 14.78
Coxa	4.75 / 4.33	5.35 / 4.81
Pé	1.43 / 1.37	1.33 / 1.29

FONTE: CLAUSER; MCCONVILLE; YOUNG (1969).

TABELA 8 - Percentual de comprimento do Segmento (a partir do próximo)

Percentual de comprimento do segmento (%)		
Segmento	**Homens**	**Mulheres**
Cabeça & Pescoço	55.0 / 50.02	55.0 / 48.41
Tronco	50.0 / 43.10	50.0 / 37.82
Antebraço	43.6 / 57.72	45.8 / 57.54
Braço	43.0 / 45.74	43.4 / 45.59
Mão	46.8 / 79.00	46.8 / 74.74
Perna	43.3 / 40.95	42.8 / 36.12
Coxa	43.4 / 43.95	41.9 / 43.52
Pé	50.0 / 44.15	50.0 / 40.14

FONTE: CLAUSER; MCCONVILLE; YOUNG (1969).

5.6.3 Por conjunto de segmentos

Essa metodologia objetiva segmentar o corpo humano de forma linear e deixar que o pesquisador atribua a cada segmento seu valor de acordo com as fórmulas a seguir:

1°**.** Definir e localizar os pontos que delimitam cada um dos segmentos que constituem o modelo segundo a FIG 60.

2°**.** Determinar os valores das coordenadas ortogonais de cada ponto (x_{di},y_{di}), coordenada distal de referência, (x_{pi},y_{pi}) coordenada proximal de referência. Tem-se tantos conjuntos de coordenadas (x_{di},y_{di}), (x_{pi},y_{pi}) quantos pontos de conexões necessários para definir os segmentos do modelo.

3°**.** Determinar a posição do centro de gravidade de cada segmento (x_{cg},y_{cg}).Ter-se-á tantas coordenadas quantos os segmentos do modelo definido. As coordenadas de posição do CG em cada segmento são calculadas a partir do comprimento do segmento. A relação algébrica entre as coordenadas da extremidade distal (x_{di},y_{di}) e a coordenada proximal (x_{pi},y_{pi}) e do valor percentual C_n considerado como localização do CG em relação ao respectivo comprimento total. Os valores das coordenadas distal e proximal são obtidos em relação a uma referência externa. Os C_n em porcentagem (%) são obtidos diretamente da tabela usada.

$$x_{cg} = x_{pi} - C_n(x_{pi} - x_{di}) \qquad \text{(eq.5.6)}$$

$$y_{cg} = y_{pi} - C_n(y_{pi} - y_{di}) \qquad \text{(eq.5.7)}$$

Sendo que:

(x_{cg},y_{cg}) = coordenadas do CG do segmento;
(x_{di},y_{di}) = coordenadas da extremidade distal de referência;
(x_{pi},y_{pi}) = coordenadas da extremidade proximal de referência;
C_n = Valores percentuais % da localização do CG de cada segmento.

4°**.** Associar a cada uma das coordenadas (x_{cg},y_{cg}) uma massa relativa m_n, isto, é de acordo com a tabela, a cada segmento do modelo é associado um valor ponderado de massa total de corpo. A associação de cada coordenada a cada massa é obtida pelo produto de dois valores (m_n,x_n) e (m_n,y_n), o indice "***n***" depende do número de segmento considerado.

5°**.** Determinar a posição do CG da totalidade do conjunto "***n***" segmento ou da totalidade do corpo, considerando que o produto dos valores de massa total e dos valores das coordenadas que se deseja determinar é igual à soma dos produtos de cada massa parcial e dos valores das respectivas coordenadas:

$$m_{cg}x_{cg} = m_1x_1 + m_2x_2 \dots m_nx_n \qquad \text{(eq.5.8)}$$
$$m_{cg}y_{cg} = m_1y_1 + m_2y_2 \dots m_ny_n \qquad \text{(eq.5.9)}$$

e portanto como $m_{cg} = 1$, fica:

$$x_{cg} = m_1x_1 + m_2x_2 \dots m_nx_n \qquad \text{(eq.5.10)}$$
$$y_{cg} = m_1y_1 + m_2y_2 \dots m_ny_n \qquad \text{(eq.5.11)}$$

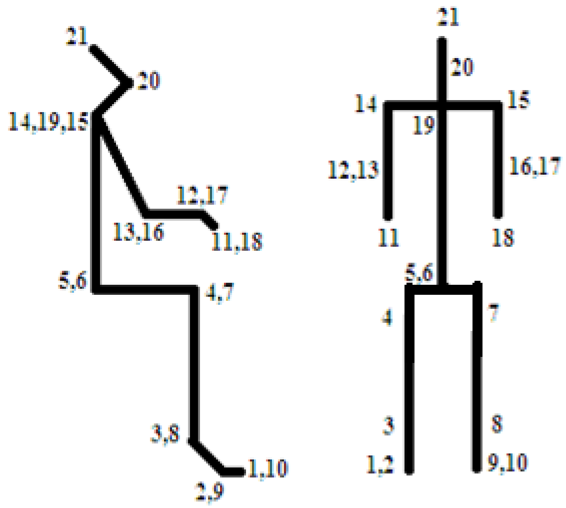


FIGURA 60 - Divisão do corpo humano em segmentos
FONTE: Dados da pesquisa.

Para isso, seria necessário introduzir a altura do usuário e o seu peso. Além disto, como a densidade do homem é diferente da mulher, era preciso definir o sexo para o sistema calcular. Pode-se imaginar como isso cria uma dependência do usuário, limitado fisicamente em relação à outra pessoa, para inserir toda essa informação.

Dessa form, decidiu-se criar um mecanismo para determinar o centro de massa do corpo humano sem a necessidade da segmentação e de tabelas, evitando cálculos intermediários que elevariam o custo do processamento e talvez, até inviabilizassem o processamento em tempo real.

5.6.4 Método da Extensiometria

Considerando-se a cadeira e o usuário como um corpo rígido, tem-se o comportamento mostrado de forma simbólica na FIG. 60 para atingir o equilíbrio.



FIGURA 61 - Inclinação da cadeira para o equilíbrio
FONTE: Dados da pesquisa.

Para se efetuar apenas o cálculo de CG da cadeira, considera-se a distância de um eixo de referência com a massa de cada parte desta e se calcula as coordenadas com as equações seguintes:

$$x_{cm} = \frac{\sum x_i \times W}{\sum W} \qquad y_{cm} = \frac{\sum y_i \times W}{\sum W} \qquad z_{cm} = \frac{\sum z_i \times W}{\sum W} \qquad \text{(eq.5.12)}$$

Sendo que:

x_{cm}, y_{cm}, z_{cm}	Representam as coordenadas do centro de massa do corpo composto;
x_i, y_i, z_i	Representam as coordenadas do centro de massa de cada parte que constitui a cadeira.
$\sum W$	É a soma dos pesos de todas as partes que constituem o corpo.

5.6.5 Método proposto para determinar o CG_{humano}

Para determinar o CG humano, desenvolveu-se uma placa para ser colocada no assento da cadeira, que leva em consideração o peso distribuído do usuário sobre ela.

Trata-se de uma placa com quatro apoios e que tem contato com os respectivos extensiômetros.

Os extensiômetros são os sensores responsáveis pelas indicações das reações. Projetou-se uma placa e, sobre esta, fixou-se uma mola com uma bola para simular o corpo humano. Além disso, fixou-se sobre esta, na parte inferior, quatro terminais que atuam sobre

quatro “strain-gages”. Cada sensor detecta uma das quatro forças para o cálculo das coordenadas do CG humano, sendo utilizado o método de equilíbrio das forças fs_1, fs_2, fs_3, fs_4 no plano XY e dos momentos destas, considerando os sensores S_1, S_2, S_3, S_4 aos pares, como indicado nas FIG. 62 e 63.

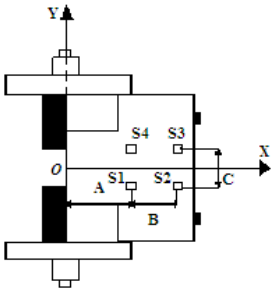


FIGURA 62 - Vista dos sensores no plano XY
FONTE: Dados da pesquisa.

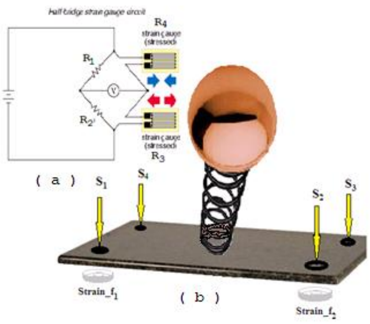
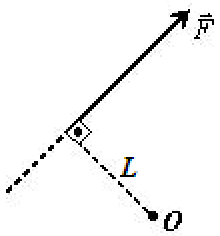


FIGURA 63 - Placa dos sensores para detectar as forças
FONTE: Dados da pesquisa.

Sendo que:

fs_1, fs_2, fs_3, fs_4	são as reações dos sensores;
A, B, C	são as distâncias;
$\vec{i}, \vec{j}, \vec{k}$	são vetores unitários.



$$M = \pm F.L \qquad \text{(eq.5.13)}$$

F – força (em Newton - N)
L – distância (em metros - m)
M – momento da força – N.m

FIGURA 64 - Momento (M) de uma força em relação ao ponto ***O***
Fonte: Dados da pesquisa

Observação: 1) *F* e *L* são perpendiculares.
2) o momento da força: + sentido anti-horário
- sentido horário

O momento de uma força em relação a um determinado ponto mede a eficiência em causar rotação de um corpo extenso em torno deste ponto. Adicionalmente pode-se definir:

5.6.6 Momento de uma força

O momento de uma força, em relação a um ponto fixo **0**, origem de um referencial inercial, é uma grandeza física, de caráter vetorial, definida pelo produto vetorial do vetor posição do ponto de aplicação da força, em relação ao ponto **0**, pela força aplicada.

5.6.7 Características do momento de uma força

Direção: perpendicular ao plano definido pelos vetores posição e força.
Sentido: dado pelas regras da mão direita ou do saca–rolhas.
Ponto de aplicação: **0**
Norma:

$$\left\|\vec{M}_0(\vec{F})\right\| = \left\|\vec{r}\right\|\left\|\vec{F}\right\|\operatorname{sen}\theta \qquad \text{(eq.5.14)}$$

O ângulo indicado é o que os vetores posição e força formam entre si. A unidade SI dessa grandeza é o N.m.

O momento de uma força tem, na rotação, um papel equivalente ao da força, na translação. O efeito de rotação produzido pela força aplicada a um corpo é medido pelo momento dessa força.

O momento de um sistema de forças, em relação a um ponto fixo **0**, em um referencial inercial, é igual à soma dos momentos das diferentes forças relativamente ao mesmo ponto.

$$\vec{M}_{sist} = \sum_{i=1}^{n} \vec{M}_i \qquad \text{(eq.5.15)}$$

5.6.8 Momento de uma força e de um sistema de forças em relação a um eixo

O momento de uma força, em relação a um eixo fixo, em um referencial inercial, é o vetor deslizante que se obtém projetando no eixo de rotação o momento da força, em relação a qualquer ponto do eixo.

O momento de uma força, relativamente a um eixo, é nulo quando a força e o eixo forem coplanares.

O momento de um sistema de forças, em relação ao eixo em torno do qual gira um corpo, é igual à soma dos momentos das forças exteriores que atuam sobre o corpo, relativamente ao eixo referido.

$$\vec{M}_{zsist} = \sum_{i=1}^{n} \vec{M}_{zi} \qquad \text{(eq.5.16)}$$

5.6.9 Momento de um binário

Um binário de forças é um sistema de duas forças, simétricas e com linhas de ação paralelas, que produzem efeito rotativo.

5.6.10 Características do momento de um binário

Direção: perpendicular ao plano do binário
Sentido: dado pelas regras da mão direita ou do saca – rolhas
Norma:

$$\left\|\vec{M}_{binario}\right\| = \left\|\vec{r}_1\right\| \left\|\vec{F}_1\right\| \text{sen } \theta + \left\|\vec{r}_2\right\| \left\|\vec{F}_2\right\| \text{sen } \theta = 2rF \text{ sen } \theta \qquad \text{(eq.5.17)}$$

Nota-se que as normas dos vetores posição dos pontos de aplicação das forças, relativamente ao ponto fixo **0**, são iguais e que as normas das forças aplicadas também o são.

Um torque devido a forças não altera o movimento de translação do corpo em que atua e produz efeito de rotação sobre o corpo em que é aplicado.

5.6.11 Cálculo para determinação do CG humano

A partir de então, calcula-se a somatória dos momentos entre as forças atuantes $\sum M_{v,w}$ e, ao se igualar a zero, obter-se-á as retas que se cruzam nas coordenadas X e Y do CG.

Para a coordenada Z, utiliza-se um inclinômetro que toma como referência o mesmo referencial inercial.

Considerando, primeiramente, a somatória dos momentos $\sum M_{1,2}=0$ em relação ao eixo de referência dos sensores S_1 *e* S_2, tem-se que:

$$f_{S1}\vec{k}\left(A\vec{\imath}+\frac{C\vec{\jmath}}{2}\right)+f_{S2}\vec{k}\left(B\vec{\imath}+\frac{C\vec{\jmath}}{2}\right)=(f_{S1}+f_{S2})\vec{k}(x\vec{\imath}-y\vec{\jmath}) \quad \text{(eq.5.18)}$$

Efetuando o produto vetorial, tem-se:

$$f_{S1}A\vec{\jmath}+\frac{f_{S1}C\vec{\imath}}{2}+f_{S2}B\vec{\jmath}+\frac{f_{S2}C\vec{\imath}}{2}=(f_{S1}+f_{S2})x\vec{\jmath}-(f_{S1}+f_{S2})y\vec{\imath} \quad \text{(eq.5.19)}$$

Colocando os termos semelhantes em evidencia, tem-se:

$$(f_{S1}+f_{S2})\frac{C\vec{\imath}}{2}+(f_{S1}A+f_{S2}B)\vec{\jmath}=-(f_{S1}+f_{S2})y\vec{\imath}+(f_{S1}+f_{S2})x\vec{\jmath} \quad \text{(eq.5.20)}$$

Fazendo $(fs_1+fs_2)=K_1$, tem-se os seguintes pontos:

$$x_1=\frac{(f_{S1}A+f_{S2}B)}{K_1} \quad \text{(eq.5.21)}$$

$$y_1=-\frac{C}{2} \quad \text{(eq.5.22)}$$

Da mesma forma anterior, faz-se as somatórias dos momentos $\sum M_{2,3}=0$ em relação ao eixo de referência dos sensores S_2 *e* S_3, obtendo-se:

$$(f_{S2}-f_{S3})\frac{C\vec{\imath}}{2}+(f_{S2}+f_{S3})B\vec{\jmath}=-(f_{S2}+f_{S3})y\vec{\imath}+(f_{S2}+f_{S3})x\vec{\jmath} \quad \text{(eq.5.23)}$$

Fazendo $fs_2-fs_3=K_2$ *e* $fs_2+fs_3=K_3$, tem-se os seguintes pontos:

$$x_2=B \quad \text{(eq.5.24)}$$

$$y_2=-\frac{K_2C}{2K_3} \quad \text{(eq.5.25)}$$

Para $\sum M_{3,4}=0$ em relação ao eixo de referência dos sensores S_3 *e* S_4, obtém-se:

$$-(f_{S3}+f_{S4})\frac{C\vec{\imath}}{2}+(f_{S3}B+f_{S4}A)=-(f_{S3}+f_{S4})y\vec{\imath}+(f_{S3}+f_{S4})x\vec{\jmath} \qquad \text{(eq.5.26)}$$

Fazendo $f_{S3}+f_{S4}=K_4$, tem-se os seguintes pontos:

$$x_3=\frac{(f_{S3}B+f_{S4}A)}{K_4} \qquad \text{(eq.5. 27)}$$

$$y_3=\frac{C}{2} \qquad \text{(eq.5.28)}$$

E, finalmente, para $\sum M_{4,1}=0$ em relação ao eixo de referencia dos sensores S4,S1, temos:

$$(f_{S1}-f_{S4})\frac{C\vec{\imath}}{2}+(f_{S1}+f_{S4})A\vec{\jmath}=-(f_{S1}+f_{S4})y\vec{\imath}+(f_{S1}+f_{S4})x\vec{\jmath} \qquad \text{(eq.5.29)}$$

Fazendo *(fs_1-fs_4)=K_5 e (fs_1+fs_4)=K_6*, tem-se os seguintes pontos:

$$x_4=A \qquad \text{(eq.5.30)}$$

$$y_4=-\frac{K_5C}{2K_6} \qquad \text{(eq.5.31)}$$

Considerando os pontos (x_1,y_1) *e* (x_3,y_3), será traçada a reta r_1:

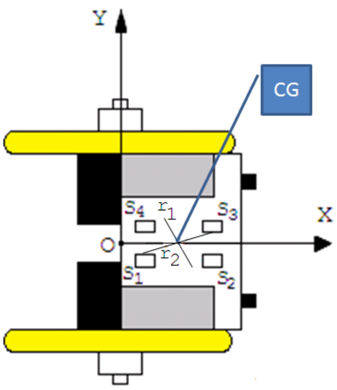


FIGURA 65 - Vista do plano XY e as retas r_1 e r_2
FONTE: Dados da pesquisa.

$$a_1=\frac{\frac{C}{2}-\frac{-C}{2}}{\frac{f_{S4}A+Bf_{S3}}{K_4}-\frac{f_{S1}A+Bf_{S2}}{K_1}} \qquad \text{(eq.5.32)}$$

$$a_1 = \frac{CK_1K_4}{f_{S4}AK_1 + f_{S3}BK_1 - f_{S1}AK_4 - f_{S2}BK_4} \quad \text{(eq.5.33)}$$

$$a_1 = \frac{CK_1K_4}{A(f_{S4}K_1 - f_{S1}K_4) + B(f_{S3}K_1 - f_{S2}K_4)} = K_7 \quad \text{(eq.5.34)}$$

A equação da reta r_1 será:

$$y_1 + \frac{C}{2} = K_7 * (x - \frac{fs_1 A + fs_2 B}{K_1}) \quad \text{(eq.5.35)}$$

Considerando os pontos *(x_2,y_2) e (x_4,y_4)*, será traçada a reta r_2:

$$a_2 = \frac{\frac{-K_5C}{2K_6} - \frac{K_2C}{2K_3}}{A - B} = K_8 \quad \text{(eq.5.36)}$$

A equação da reta r_2 será:

$$y_2 + \frac{K_2C}{2K_3} = K_8(x - B) \quad \text{(eq.5.37)}$$

Isolando o y e substituindo a reta r_2 em r_1, obtêm-se as coordenadas do CG no plano xy.

$$x_{CG} = \frac{K_1K_2C + 2K_1K_3K_8B + K_1K_3C - 2K_3K_7(f_{S1}A + f_{S2}B)}{2K_1K_3(K_8 - K_7)} \quad \text{(eq.5.38)}$$

$$y_{CG} = -\frac{K_2C}{2K_1} + K_8 \frac{(K_1K_2C + 2K_1K_3K_8B + K_1K_3C - 2K_3K_7(f_{S1}A + f_{S2}B) - B)}{(2K_1K_3(K_8 - K_7))} \quad \text{(eq.5.39)}$$

Diante da dificuldade encontrada para o cálculo da altura do CG somente com os sensores existentes, optou-se pelo uso de um inclinômetro para determinação do ângulo θ entre a cadeira e o chão. Com isso, será determinada a altura do CG.

Considerando as somatórias dos momentos $\sum M = o$ em relação ao eixo de referência no plano xz, tem-se:

$$f_{S1}(\cos\theta\,\vec{\imath} - \sin\theta\,\vec{\jmath})A(\cos\theta\,\vec{\imath} + \sin\theta\,\vec{\jmath}) + f_{S2}(\cos\theta\,\vec{\imath} - \sin\theta\,\vec{\jmath})B(\cos\theta\,\vec{\imath} - \sin\theta\,\vec{\jmath}) + f_{S3}(\cos\theta\,\vec{\imath} - \sin\theta\,\vec{\jmath})B(\cos\theta\,\vec{\imath} + \sin\theta\,\vec{\jmath}) + f_{S4}(\cos\theta\,\vec{\imath} - \sin\theta\,\vec{\jmath})A(\cos\theta\,\vec{\imath} + \sin\theta\,\vec{\jmath}) = P(\cos\theta\,\vec{\imath} - \sin\theta\,\vec{\jmath})\big(x(\cos\theta\,\vec{\imath} + \sin\theta\,\vec{\jmath}) + Z\vec{k}\big)$$

(eq.5.40)

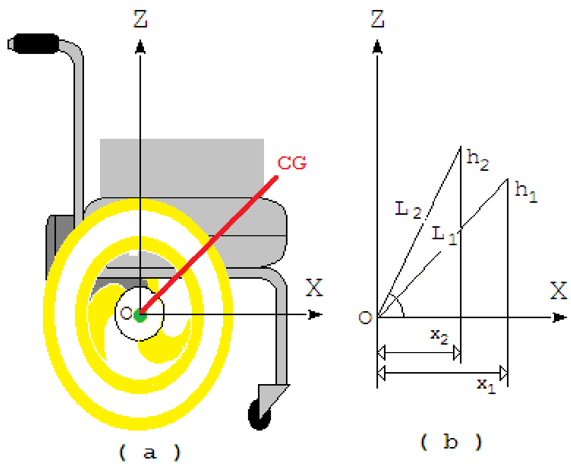


FIGURA 48 - (a) Vista do plano XZ (b) Variação da altura do CG
FONTE: Dados da pesquisa.

Efetuando o produto vetorial, tem-se:

$$(f_{S1}A + f_{S2}B + f_{S3}B + f_{S4}A)\big(2(\cos\theta\sin\theta)\big)\vec{\jmath} = 2P(\cos\theta\sin\theta)\vec{\jmath} + PZ\cos\theta\,\vec{\jmath}$$

(eq.5.41)

Com isso, obtém-se a coordenada z do CG, a sua altura:

$$z_{CG} = \frac{\big((f_{S1}+f_{S4})A+(f_{S2}+f_{S3})B\big)2(\cos\theta\sin\theta) - 2Px_{CG}(\cos\theta\sin\theta)}{P\cos\theta} \qquad \text{(eq.5.42)}$$

Cálculo alternativo da altura do CG:

Considerando:

x_1 *e* x_2 = coordenadas do eixo X;
d = distância do eixo de referencia ao CG;
H_1 *e* H_2 = alturas do CG;
α_1 e $\Delta\alpha$ = ângulo entre o plano da cadeira e o solo.

$$x_1^2 + H_1^2 = d^2 \qquad \text{(eq.5.43)}$$

$$x_2^2 + H_2^2 = d^2 \qquad \text{(eq.5.44)}$$

Igualando as duas expressões anteriores, tem-se:

$$x_1^2 + H_1^2 = x_2^2 + H_2^2 \qquad \text{(eq.5.45)}$$

Sendo que:

$$H_1 = d\sin(\theta) \qquad \text{(eq.5.46)}$$

$$H_2 = d\sin(\theta + \Delta\theta) \qquad \text{(eq.5.47)}$$

Mas:

$$d\cos\theta = x_1 \qquad \text{(eq.5.48)}$$

$$d\cos(\theta + \Delta\theta) = x_2 \qquad \text{(eq.5.49)}$$

Dividindo a primeira pela segunda:

$$\frac{\cos\theta}{\cos(\theta+\Delta\theta)} = \frac{x_1}{x_2} \qquad \text{(eq.5.50)}$$

Sendo que:

$$d = \frac{x_1}{\cos\theta} \qquad \text{(eq.5.51)}$$

$$H_1 = \sqrt{d^2 - x_1^2} \qquad \text{(eq.5.52)}$$

6 SIMULAÇÕES E ANÁLISE DOS RESULTADOS

Anteriormente, os valores da saída relativos à distância correspondem praticamente ao valor do estado velocidade, no caso do ângulo, é apenas aproximado. Em caso de algum distúrbio como atritos e movimentações do usuário, não corresponderia como pode ser observado na bibliografia.

Foram feitas as alterações necessárias no modelo de modo a fazer com que a saída corresponda à média dos valores dos estados do sistema. Adiante, a simulação em Matlab® do sistema, mostrando as saídas em cores verde (inclinação angular) e azul (posição no plano)

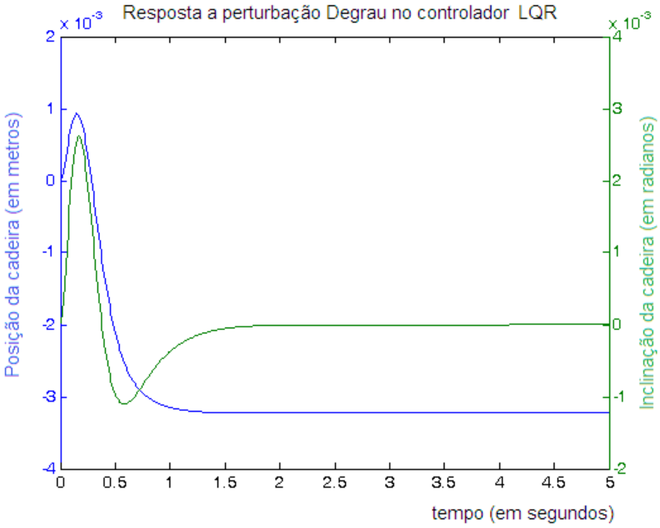


FIGURA 49 - A resposta do sistema controlado para entrada degrau que simula uma perturbação no sistema com o controle LQR
FONTE: Dados da pesquisa.

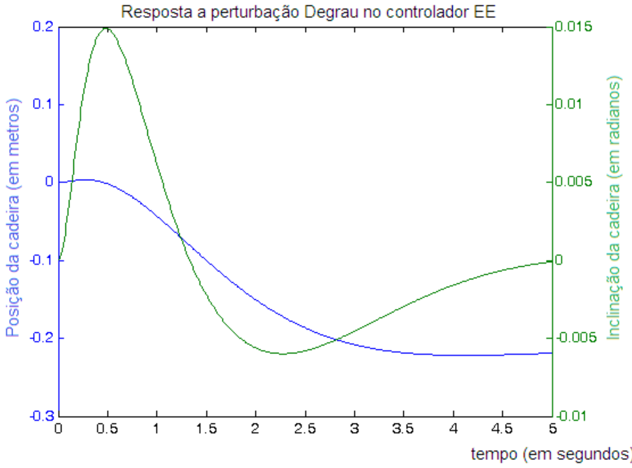


FIGURA 50 - Resposta à perturbação degrau de um controlador moderno de estados realimentado
FONTE: Dados da pesquisa.

A resposta deste tipo de controle demonstra aquilo que já era conhecido do controle convencional na bibliografia e atende as exigências necessárias dos controladores.

6.1 Por Lógica Nebulosa (“Fuzzy”)

Definido o controlador “fuzzy”, criou-se uma estrutura que pudesse simular e testar se este seria capaz de controlar a cadeira em torno do ponto desejado. Para esses procedimentos de simulação, foram utilizadas as ferramentas de matemática da “MathWorks”, por meio dos programas MatLab e Simulink, que fornecem, conforme a versão, um grau de precisão aceitável para validação e posterior aplicação. Este diagrama pode ser visto na FIG. 68.

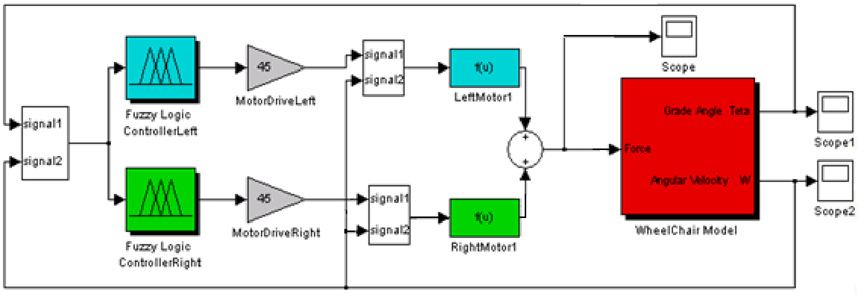


FIGURA 69 - Diagrama do sistema da cadeira com o controlador “fuzzy” sem perturbações
FONTE: Dados da pesquisa.

Os resultados obtidos na simulação do esquema da FIG. 69 demonstram que o controlador “fuzzy” atua corretamente e a oscilação embora excessiva, inicialmente não esperada, vai sendo gradualmente corrigida dentro de um tempo de amortecimento muito bom (5 segundos), levando ao equilíbrio, ou seja, ao ângulo Theta esperado de 90º. Este ângulo Theta parte daquele valor de 69°, conforme observado no segundo parágrafo do item 2 (modelo matemático), e se refere à posição inicial do CG ou centro de gravidade da cadeira com o seu usuário. As formas de onda podem ser vistas na FIG. 70.

Como pode ser visto na FIG. 70, o controlador atua com uma ação variável em módulo e sentido, até o momento em que o CG total (centro de gravidade resultante) da cadeira atinge o equilíbrio, com as suas variáveis “fuzzy”, ângulo, velocidade angular e saída atingindo os valores EQ, ZE e ZE, respectivamente. Mas pode também ser observado que este controlador oscila além do desejado o que causará um desconforto ao usuário tetraplégico. De acordo com práticas anteriores deste tipo de projeto, este problema pode ser corrigido com uma base de regras mais rica de condições o que leva a um controle de sintonia mais fina.

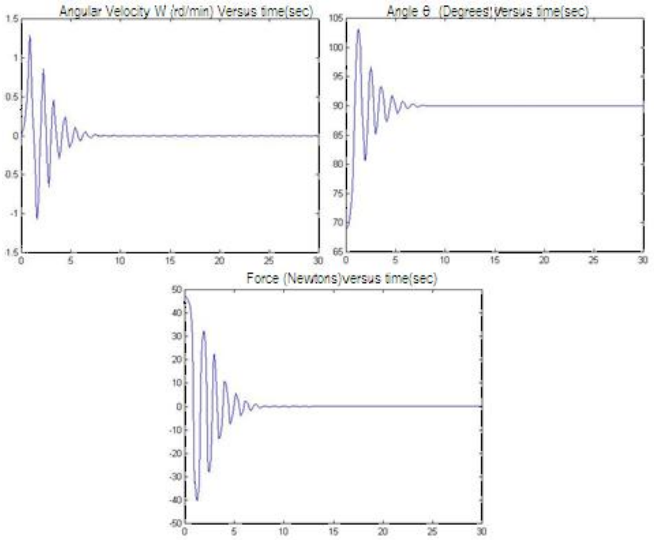


FIGURA 70 - Resultados da simulação sem perturbação com a distância **d** do CG fixa
FONTE: Dados da pesquisa.

Para verificar o comportamento do sistema diante de vários tipos de perturbação e comprovar se a estabilidade do controlador era aceitável, foi montado um novo esquema (incluindo parâmetros físicos com seus valores mais reais), conforme pode ser visto na FIG. 71 que contemplasse essa situação, de forma que, a partir de agora, passa a existir uma perturbação externa no instante de tempo t = 30s, correspondente a um intervalo apreciável após o controlador ter levado o sistema ao equilibrio. Além dessa perturbação, outra, de origem interna, gerada pelo movimento do usuário, afeta o posicionamento do CG pela modificação de sua distância **d** ao referencial inercial.

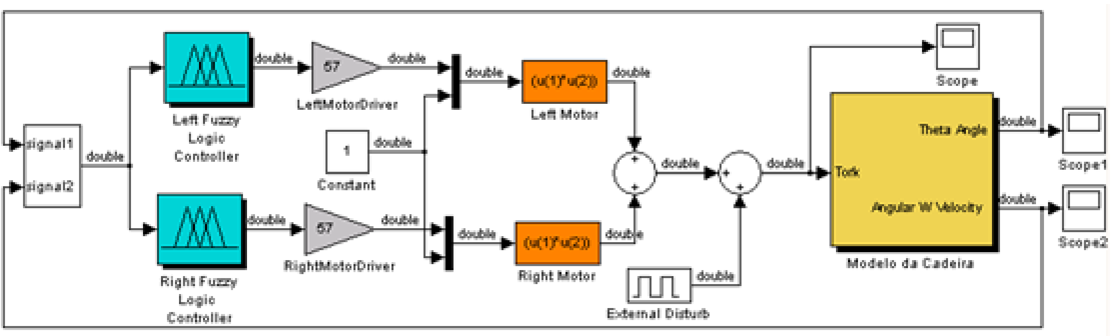


FIGURA 51 - O controlador “fuzzy” face à existência de uma perturbação externa
FONTE: Dados da pesquisa.

Essas perturbações foram simuladas conforme os dois diagramas em blocos do Simulink® apresentados a seguir com as suas respectivas saídas e podem ser vistos na FIG. 72 e FIG. 73.

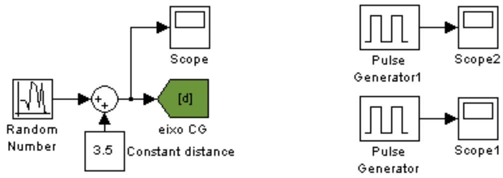


FIGURA 72 - O controlador “fuzzy” face à existência de perturbações interna e externa
FONTE: Dados da pesquisa.

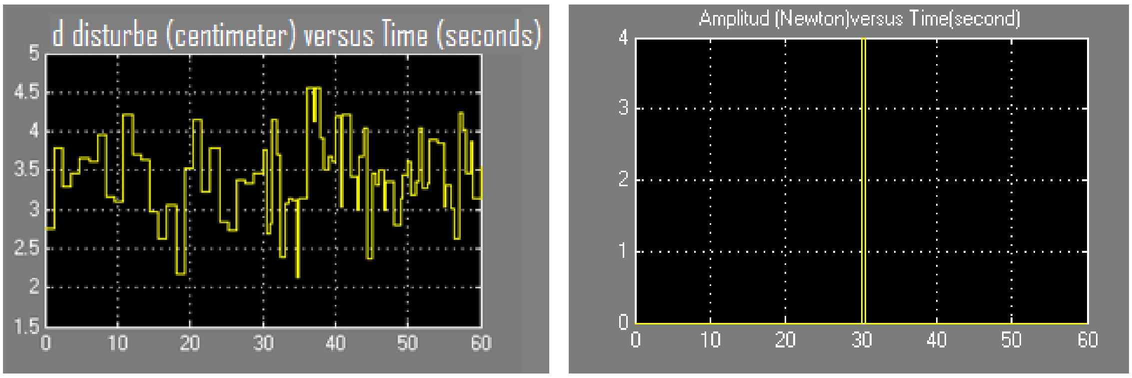


FIGURA 73 - As perturbações randômicas internas (scope) e colisões externas (scope1)
FONTE: Dados da pesquisa.

Os resultados obtidos na simulação dos esquemas da FIG. 71, os quais apresentam as perturbações da FIG. 73, podem ser vistos na FIG. 74, a partir da qual se percebe que o controle atua continuamente e o sistema não se desequilibra diante das diversas perturbações combinadas.

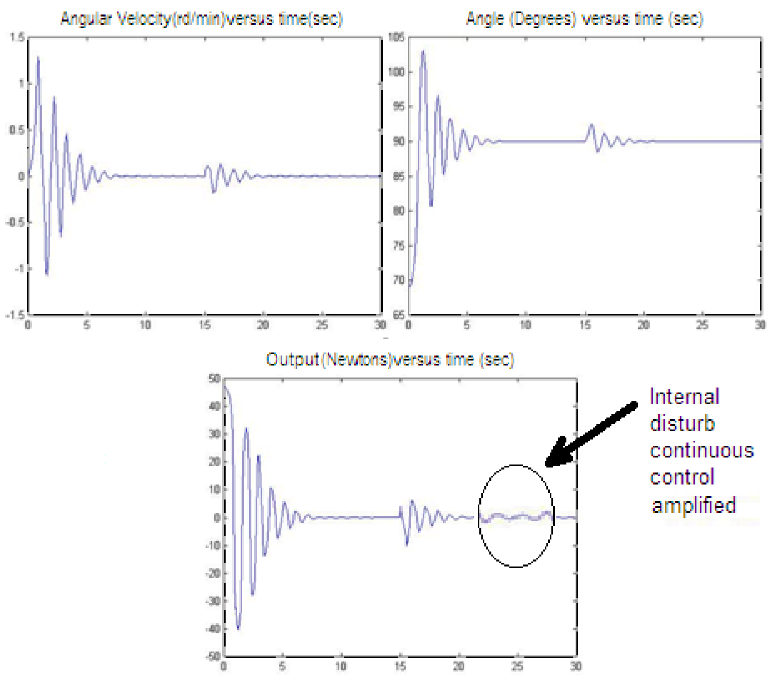


FIGURA 74 - Perturbações severas e diversas levando a uma distância **d** do CG variável
FONTE: Dados da pesquisa.

De fato, considera-se que a perturbação interna, devido à movimentação do usuário, é randômica e contínua como pode ser observado na lente de ampliação da FIG. 73, levando o controlador “fuzzy”, por sua vez, a agir continuamente. A oscilação que aparece de forma amplificada na lente gráfica virtual demonstra esse tipo de ação de controle.

Os resultados obtidos no trabalho apresentado têm por finalidade avaliar o desempenho de um controlador Nebuloso e compará-lo ao controlador Moderno, baseado em Espaço de Estado, e justificar que a decisão de se adotar controladores baseados na lógica “fuzzy” pode ser viável.

As simulações dos diversos controladores e daquela do controlador nebuloso, que mostra a resposta deste através do método de defuzzificação Mandani, com todas as variáveis de estado chegando a zero em aproximadamente 20 segundos de simulação a um custo computacional estimado de $O(n^3)$, apontam para uma ampla possibilidade de se empregar esse tipo de controle, mesmo comparado ao tempo menor para a estabilidade pelo controlador pelo Espaço de Estados a um custo computacional de $O(n^2)$.

Apesar da visível vantagem inicial do controlador por Espaço de Estados, essa situação começa a se reverter quando se introduz a figura do usuário com a massa diferente e movimentações que causam a mudança do CG. Isso implica em conflito com os parâmetros originais do projeto do controlador moderno por Espaço de Estado. Para resgatar sua estabilidade, o controlador por Espaço de Estado deveria se adaptar às mudanças causadas pelo usuário, o que implica, como se viu, em um cálculo mais elaborado da nova posição do CG, a um custo computacional maior na ordem de $O(n^4)$.

O calculo do CG, em tempo real, para atualização do controlador moderno, torna-o adaptativo e estável, mas menos competitivo em relação ao controlador “fuzzy”. Este, por sua vez, não necessita dessa atualização, pois, como se viu, sob perturbações severas, ele foi capaz de estimar a ação necessária para manter a estabilidade com o mesmo custo de $O(n^3)$. Isso se deve ao fato de que as funções de pertinência foram bem dimensionadas para o processo de fuzzyficação.

Os resultados das simulações levam à conclusão de que o controlador “fuzzy” atendeu às exigências de controle da cadeira em questão, corrigindo as perturbações, mesmo nos casos mais severos, como as perturbações randômicas (internas e externas) e contínuas (internas) combinadas.

As simulações dos diferentes controladores mostram uma mesma resposta, com todas as variáveis de estado chegando a zero em aproximadamente 5 segundos, mas pelo método de Espaço de Estados, apresentou menores oscilações (mais confortável).

Quando ocorre variação do CG devido às perturbações, pela natureza evasiva do controlador “fuzzy”, ele aceita e controla (robusto), mas o controle moderno encontra um limite por ser determinístico e já não controla como originalmente necessitando adaptações.

Os resultados obtidos não tiveram por finalidade avaliar o desempenho de um controlador Nebuloso e compará-lo ao controlador Moderno, baseado em Espaço de Estado, apenas a de justificar que a decisão de se adotar controladores baseados na lógica “fuzzy” pode ser viável se o universo de conhecimento é dominado.

O que ficou claro pelas simulações é que o sistema, em alguns casos, com o controle feito pelo método de Espaço de Estados, ainda apresentou maiores oscilações, chegando a um gasto computacional na ordem de $O(n^4)$, o que deixa explícito uma dificuldade maior no processamento dos dados.

Nos testes realizados com o Controle Ótimo, sem alteração das condições de projeto, através de várias simulações, após se obter um bom conjunto de ganhos, o ângulo da Cadeira de Rodas atingiu o valor zero e a velocidade do carrinho, um valor muito próximo a zero, como demostrado anteriormente. Os valores para o Espaço de Estado, dentro de um tempo de simulação de 100 segundos, têm um deslocamento da cadeira superior ao do controlador “fuzzy”.

Mesmo com as perturbações de origem externa (forças que alteram a ação do controlador) e internas (movimentação do usuário na cadeira afetando o **CG** e sua distância **d** em relação ao referencial inercial **O**) existentes na planta, como se viu, o controlador “fuzzy” atuou de forma satisfatória e fez com que o sistema estabilizasse novamente, comprovando, assim, a eficácia desse tipo de controlador baseado em conhecimento.

7 CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho foram apresentados resultados obtidos através da avaliação de simulações do controlador "Fuzzy" para o problema de uma cadeira de rodas robótica e equilibrista, com base no princípio do pêndulo invertido, comparando-o com os resultados do controlador Ótimo. Todos os experimentos foram realizados utilizando-se a formulação matemática da dinâmica de comportamento do pêndulo descrita ao longo deste estudo onde se tomou todo o cuidado de evidenciar a importância da inércia nas equações da dinâmica.

O desenvolvimento de controladores para sistemas dinâmicos não lineares compreende um desafio. Quando se procura controlar tais sistemas, a especificação de um controlador "Fuzzy" leva uma vantagem considerável sobre o Ótimo, pois tem uma abordagem mais intuitiva, baseada em variáveis e regras linguísticas. Todavia, quando a complexidade do problema aumenta, sua dimensão cresce proporcionalmente, dificultando, de certa forma, a definição de regras Fuzzy.

Esse sistema, da maneira que se apresenta, pode ser aplicado a outros domínios do conhecimento, desde que sejam modelados com outras entradas e saídas.

A proposta atual para a implementação deste trabalho é a interação entre qualquer usuário e a cadeira de rodas, o que pode ser feito através de comandos vocais de uma interface baseada em redes neurais capaz de personalizar tal interpretação. Na implementação prática, planeja-se que o movimento da cadeira se dê a partir de um comando preliminar vocal interpretado pela rede neural que, tomando a ação desejada, mantêm-se nela até que seja dada uma contra ordem ou uma nova ação automática pela fusão sensorial.

A soma dessas contribuições e as suas repercussões para o atendimento aos desabilitados físicos será bem ampla e acredita-se muito bem aceita pela comunidade de usuários com problemas de tetraplegia, sendo que, adicionalmente, pode-se admitir:

a) segurança: em função das condições das calçadas, como pedras soltas, tipos diferentes de piso, o cadeirante fica sujeito a acidentes pela falta de controle da cadeira de rodas convencional. A cadeira de rodas equilibrista irá contar com um controle eficaz para evitar colisões, proporcionando maior conforto e segurança ao cadeirante, além da possibilidade de locomoção mais segura, facilitada e, ainda, satisfação e sentimento de integração social;

b) melhoria da autoestima: nada mais importante para um cadeirante que uma cadeira de rodas, que é, praticamente, uma extensão do seu corpo. Portanto, uma cadeira que proporcione mais segurança e possibilidade de locomoção facilitada torna a pessoa com deficiência mais independente e melhora sua autoestima;
c) acessibilidade: Segundo a Organização Mundial de Saúde (OMS), acessibilidade está relacionada a fornecer condição para utilização, com segurança e autonomia, total ou assistida, dos espaços, mobiliários e equipamentos urbanos, das edificações, dos serviços de transporte e dos dispositivos, sistemas e meios de comunicação e informação, por pessoa portadora de deficiência ou com mobilidade reduzida. Porém, para o deficiente físico, acessibilidade assumiu dimensão mais ampla, pois diz respeito à qualidade ou falta de qualidade de vida. A cadeira de rodas não é somente um meio de locomoção, é um acessório de inclusão social. Portanto, as considerações apresentadas neste estudo evidenciam a importância da continuidade do projeto de forma a ajudar pessoas com incapacidades ou deficiências a executarem atividades do cotidiano com mais segurança, trazendo, com isso, melhoria das condições de vida dessas pessoas com necessidades especiais.

Outra proposta já em andamento é a de aperfeiçoamento dos sensores dessa cadeira de rodas, o que implica em uma determinação melhor da modificação do posicionamento do **CG** do sistema. Isso corrige a alteração da inércia e, consequentemente, da dinâmica da cadeira que, sob um programa integrado de controle, será eficaz, podendo até mesmo redimensionar os limites do controlador “fuzzy”, tornando-o adaptativo.

O deslocamento de uma cadeira de rodas para desabilitados, com a sua inclinação mantida em relação ao plano horizontal, com giros para a esquerda e direita, é sempre instável e sujeito a ruídos. Isso justifica estudos para se introduzir novos conceitos de projeto a fim de dotar esse meio de locomoção de um comportamento mais adequado e adaptativo, possibilitando que, no futuro, seja até preditivo.

Os diversos elementos no conjunto do projeto, determinados pelo projetista, inclusive a montagem mecânica básica atenderam o objetivo pretendido do uso da inteligência artificial para o controle de um robô, neste caso, a cadeira. A meta só não não foi atingida, pelas restrições iniciais impostas nesta tese.

Ficou claro por todas simulações realizadas, que o controle “fuzzy” é viável, porém para maior eficiência e eficácia do sistema, pode ser adaptado pela ampliação das regras e modificações nas funções de pertinência.

Sistemas dinâmicos não lineares são um desafio e um controlador “Fuzzy” leva uma vantagem considerável sobre o Moderno por espaço de estados, pois tem uma abordagem mais intuitiva, baseada em variáveis e regras linguísticas. Ou seja, seu custo de desenvolvimento é muito menor do que aquele feito para o controle convencional.

Deve-se levar ainda em consideração o custo computacional estimado em **$O(n^x)$**. Conforme dissemos anteriormente, uma definição mais detalhada está no apêndice 2.

A soma dessas contribuições metodológicas e as suas repercussões para o atendimento aos desabilitados físicos será bem ampla e acredita-se muito bem aceita pela comunidade de medicina e pelos usuários com problemas de tetraplegia.

Pretende-se doar tal estudo a instituições e/ou fundações internacionais interessadas, para fins filantrópicos, caso não exista interesse comercial que financie e apoie os desabilitados físicos, acompanhando todo o seu desenvolvimento e ajudando no que for possível.

Os resultados das simulações levam à conclusão de que o controlador “fuzzy” atendeu às exigências de controle da cadeira em questão, corrigindo as perturbações, mesmo nos casos mais severos, como as perturbações randômicas (internas e externas) e contínuas (internas) combinadas.

REFERÊNCIAS BIBLIOGRÁFICAS[4]

ABRANTES, J. M. *Fundamentos e elementos de análise em biomecânica do movimento humano*. Lisboa, Portugal: Universidade Lusófona, 2008.

AGUIAR, A. P.; PASCOAL, A. Stabilization of the extended nonholonomic double integrator via logic-based hybrid control. In: *SYROCO, 6.*, 2000, Viena. *Proceedings of IFAC Symposium on Robot Control.* Austria, 2000.

AHMAD, S.; SIDDIQUE, N. H. A modular fuzzy control approach for two-wheeled wheelchair. *Journal of Intelligent & Robotic Systems,* v. 64, n. 3/4, 2011.

ALEXANDER, J. C.; MADDOCKS, J. H. On the kinematics of wheeled mobile robots. *International Journal of Robotic Research,* v. 8, n. 5, p. 15-27, Oct. 1989.

AL-THAHAB, A. Q. Controlled of mobile robots by using speech recognition. *Journal of Babylon University - Pure and Applied Sciences*, v. 19, n. 3, p. 201, 2011.

ALVARENGA, F. B. *Desenvolvimento de sistemas de motorização alternativa para cadeiras de rodas convencionais.* 2002. Dissertação (Mestrado em Engenharia Mecânica) – Universidade Estadual de Campinas, Faculdade de Engenharia Mecânica, UNICAMP, Campinas, 2002.

AMORIM, B. M. P. *Uma contribuição crítica para o redesenho de cadeiras de rodas adaptadas para crianças e adolescentes com paralisia cerebral*. 2009. 108f. Dissertação (Mestrado) - Universidade Federal do Rio Grande do Norte. Centro de Tecnologia. Programa de Pós-Graduação em Engenharia de Produção, 2009.

ARNOL'D, V. I.; NOVIKOV, S. P. (Eds.). Dynamical systems VII: integrable systems, nonholonomic dynamical systems. *In:* ENCYCLOPAEDIA of Mathematical Sciences. New York: Springer-Verlag, 1994. v. 16; n. 7.

ASTROM, K. I.; WITTENMARK, B. CompIIoter-conlTolled systems: theory and design. Englewood Cliffs, N. Iersey: Prentice Hall, 1990.

BEALE, R; JACKSON, T. *Neural computing:* an introduction. Bristol: Adam Hilger, 1990.

BECKER, M. *Aplicação de tecnologias assistivas e técnicas de controle em cadeiras de rodas inteligentes.* 2000. Tese (Doutorado em Engenharia Mecanica) – Universidade Estadual de Campinas, Faculdade de Engenharia Mecânica, UNICAMP, Campinas, 2000.

BOISSONNAT, J.; AREZO A.; LEBLOND, L. Shortest paths of bounded curvature in the plane. *In:* CONF. ROBOTICS AND AUTOMATION, 1992, Nice, France. *Proceedings…* Nice, May 1992. v. 3. p. 2315-2320.

BORENSTEIN, J.; KOREN, Y. A mobile platform for nursing robots. I*EEE Trans. Ind. Electron.*, v. IE-32, p. 158-165, June 1985.

[4] Lista de referências citadas diretamente no texto e de fontes utilizadas nas leituras para a concepção do conteúdo.

BORENSTEIN, J.; KOREN, Y. Motion control analysis of a mobile robot. ASME J. DynamicSyst., v. 109, p. 158-165, Jun. 1987.

BROCKEN, R. W. Asymptotic stability and feedback stabilization. *In:* BROCKEN, R. W.; SUSSMAN, M. (Eds). *Differential geometric control theory*. Boston: Birkhauser, 1983. p. 181-208.

CARBONELL, J. G. *Paradigms for machine learning.* [*s. l. : s. n.*], 1989. v. 40. p. 1-9.

CARNEGIE-MELLON. *Control tutorials for Matlab*. Ohio State: University of Michigan, 2001.

CHAN, M. World health report: research for universal health coverage. [s.l.:s.n], 3 March, 2013.

CHEN, C.-T.; PHAM, H.-V. Enhanced development and stability analysis of a new stair-climbing robotic wheelchair. *In:* IEEE INTERNATIONAL CONFERENCE ON ADVANCED ROBOTICS AND ITS SOCIAL IMPACTS TAIPEI, Taiwan, Aug. 23-25, 2008. *Proceedings…* Taiwan, 2008.

CHIANG, R.; SAFODOV, M. Robust Conno! Toolboxfor use with MatlabP. Natick, Mass.: The MathWorks Inc., 1992.

CLAUSER, C. E.; MCCONVILLE, J. T.; YOUNG, J. W. *Weigth, volume and center of mass of segments of the human body*. Ohio, United States: Nasa; Aerospace Medical Research Laboratory; Aerospace Medical Division; Wright-Patterson Air Forces Base, August 1969.

COSTA, P. G. Identificação, modelização e controlo de um veículo móvel autônomo. 1995. 165f. Dissertação (Mestrado em Engenharia Electrotécnica e de Computadores) – Faculdade de Engenharia da Universidade do Porto, Porto, mar. 1995.

COX, I. B. An experiment in guidance and navigation of an autonomous robot vehicle. *IEEE Trans. Robotics and Automation.*, v. 7, n. 2, p. 193-204, Apr. 1991.

COX, L. L.; NELSON, W. L. Local path control for an autonomous vehicle. *In*: IEEE INTERN. CON! ON ROBOTICS AND AUTOMATION., Apr. 1988, Philadelphia. Pennsylvania. *Proceedings…* Pennsylvania, 1988. p. 1504-1510.

CRUZ, P. P.; FIGUEROA, F. P. R. *LABVIEW*: neuro-fuzzy controller theory and application. Book Title Intelligent Control Systems with LabVIEW™. London: Copyright Holder,Springer-Verlag London, 2010.

CURRAN, A.; KYRIAKOPOULOS, K. J. A sensor-based self-localization and navigation scheme for mobile robots. *Journal of Robotic Systems*, v. 12, n. 3, p. 163-176, 1995.

DEMUTH, H.; BEALE, M. *Nonlinear system identification*. Matlab: Neural Network Toolbox User's Guide, version 3.0. Natick,MA: The Mathworks inc., 1998.

DEWAN, S. B.; STRAUGHEN, A. Power semiconductor circuits. New York: John Wiley and Sons, 1984.

DEYLE, T. iBOT Discontinued -- Unfortunate for the Disabled but Perhaps a Budding Robotics Opportunity? February 11, 2009.

DUBOIS, D.; PRADE, H. *Fuzzy sets and systems*. New York: Academic Press, 1980. cap. 2. p 191-192.

DUPONT, P., YAMAJAKO, S. P. Stability of rigid-body dynamics with sliding frictional contacts. *In:* IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION, 1996, Minneapolis, MN, April 1996. *Proceedings…* Minneapolis, 1996. p. 378-384.

DUPONT, P.; DUNLAP, E.P. Friction modeling and PD compensation at very low velocities. Transactions of *ASME*, v. 8, p. 117, 1995.

EDGEWICK, R. ***Algorithms in C:*** 3. ed. Reading: Addison-Wesley, 2004. parts 1-4.

ELLIS, J. R. *Vehicle dynamics*. London, UK: London Business Books, 1969.

ESPINOSA, F. *et al.* Advanced and intelligent control techniques applied to the drive control and path tracking systems on a robotic wheelchair. *Autonomous Robots*, v. 11, n. 2, p. 137-48, 2001.

FERNANDES, C.; GURVITS, L.; LI, Z. Near-optimal nonholonomic motion planning for a system of coupled rigid bodies. *IEEE Transactions on Automatic Control,* v. 39, n. 3, p. 450-463, 1994.

FIGUEIREDO, L. C.; JOTA, F. G. Introdução ao controle de sistemas não-holonômicos. *Revista da Sociedade Brasileira de Automática*, p. 243-268, , 2003.

FINSCHI, L. *An implementation of the levenberg-marquardt algorithm*. Zürich: Institut für Operations Research, 1996.

FLEURY, S. *et al.* Primitives for smoothing mobile robot trajectories. Toulouse, France: LAAS TeclL, Sep. 1992. Report 92355.

FLYNN, A.; JONES, J. L. *Mobile robots wellesley*. Mass.: A K Peters, 1993.

FREEMAN, J. A; SKAPURA, D. M. *Neural networks algorithms, applications and programmming techniques.* [*s. l.*]: Addelfon-Wefley, 1992. p. 89-106.

GALVÃO, C. R. C. *Análise crítica dos produtos de mobilidade sentada - cadeiras de rodas - utilizados por crianças e adolescentes com paralisia cerebral em Natal/RN e outros municípios do Rio Grande do Norte*. 2006. 100f. Dissertação (Mestrado em Engenharia de Produção) – Centro de Tecnologia, Programa de Pós-Graduação de Engenharia de Produção, Universidade Federal do Rio Grande do Norte, 2006.

GASÓS, J.; ALMEIDA, A. R. de. Uncertainty representation for mobile robots: perception, modeling and navigation in unknown environments. *Fuzzy Sets and Systems Journal*, n. 107, p. 1-24, 1999.

GHAROONI, S. C.; AWADA, B.; TOKHI, M. O. *Modeling and control of upright lifting wheelchair*. In: INTERNATIONAL CONFERENCE ON CLIMBING AND WALKING

ROBOTS AND THE SUPPORT TECHNOLOGIES FOR MOBILE MACHINES (CLAWAR 2005), 8., 2006, *Proceedings…* [*s.l.: s.n.*], 2006. p. 969-976.

GHOSAL, A.; BALAKRISHNA, R. Modelling of slip for wheeled mobile robots. *IEEE Trans. Robotics and Automation*, v. 11, DO. 1, p. 126-132, Feb. 1995.

GIRALT, G.; CHATILA, R.; VAISSET, M. An integrated navigation and motion control system for autonomous multisensory mobile robots. *In:* BRADY, M.; PAUL, R. (Eds.). *Robotics research:* the first international symposium. Cambridge, MA: The MIT Press,1984. p. 191-214.

GIRALT, G.; SOBEK, R.; CHATILA, R. A multilevel planning and navigation system for a mobile robots: a first approach to HILARE. *In:* INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, 6., 1979, Tokio, Japan. *Proceedings…* Tokio, 1979. p. 335-337.

GOLDSTEIN, H. *Classical mechanics reading.* Mass.: Addinson-Wesley, 1980.

GOODWIN, G.; PAYNE, R. Dynamic system identification. New York: Academic Press. 1977.

GRACE, A. *et al. Control system toolbox/or use with Matlab*. Natick, Mass.: The MathWorks, 1992.

GRASSI, V. *Arquitetura híbrida para robôs móveis baseada em funções de navegação com interação humana*. 2006. Tese (Doutorado em Engenharia Elétrica) – Departamento de Engenharia Elétrica, Escola de Engenharia de São Carlos, USP, São Carlos, 2006.

GRAUPE, D. *Identification 0/ systems.* Malabar, Florida: Robert Krieger Publishing Company, 1976.

GUSTAFSSON, F. Slip-based estimation of tire - road friction. *Automatica*, v. 33, n. 6, p. 1087-1099, 1997.

HAMANAKA, M. H. M. O. *Projeto e desenvolvimento de circuito de controle para cadeira de rodas*. 2002. Dissertação (Mestrado em Engenharia Elétrica) - Faculdade de Engenharia Elétrica e de Computação (FEEC), 2002.

HANSELMAN, D.; LITTLEFIELD, B. *Matlab 5:* guia do usuário. São Paulo: Makron Books do Brasil, 1999.

HARMON A. An Inventor Unveils His Mysterious Personal Transportation Device.The New York Times.December 3, 2001.

HAUSER, M. W.; SANTOS, C. B. *Planilha didática para determinação do centro de gravidade do corpo humano pelo método da segmentação*. Ponta Grossa, PR, Brasil: Universidade Estadual de Ponta Grossa, 2002.

HAYKIN, S. *Neural networks:* a comprehensive foundation. New Jersey, USA: Prentice Hall,1999.

HEBB, D. O. *The organization of behavior*. New York: John-Wiley& Sons Inc, 1949.

HECHT-NIELSEN, R. Applications of counterpropagation networks. *Neural Networks Journal*, p. 131-140, 1988.

HIBBELER, R. C. *Estática:* mecânica para engenharia. 12 ed. São Paulo: Pearson, 2011.

HINTON, G. *Connectionist learning procedures*. [*s. l.*]: Computer Science Department, Carnegie-Melon University, 1987.

INDELICATO, A. *Estrutura para o reconhecimento de referências geométricas utilizando ultra-som e lógica nebulosa*. 1997. Dissertação (Mestrado em Engenharia Elétrica) – Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal do Espírito Santo, Vitória, Espírito Santo, Brasil, 1997.

INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA (IBGE). IBGE e CORDE abrem encontro internacional de estatísticas sobre pessoas com deficiência. Rio de Janeiro, 2005. Disponível em: <http://www.ibge.gov.br>. Acesso em: 21 set. 2013.

JAMSHIDI, M.; TAROKH, M.; SHAFAI, B. Computer-aided analysis and tksign 0/ linear control systems. Englewood Cliffs, N. Jersey: Prentice Hall, 1992.

JONES, W. P.; HOSKINS, J. Back-propagation: a generalized delta learning rule. *BYTE*, v. 12, n. 11, p. 155-162, October 1987.

KATSUHIKO, O. *Engenharia de controle moderno.* 5. ed. São Paulo: Pearson Education BR, 1993.

KLAFTER, T. R. C.; NEGIN, M. Robotic engineering: an integrated approach. Englewood Cliffs, N. Jersey: Prentice Hall, 1989.

KOLMANOVSKY, I.; MCCLAMROCH, N. H. Developments in nonholonomic control problems. *IEEE Trans. on Control Systems*, v. 15, n. 6, p. 20-36, 1995.

KOMIYA, K. *et al.* Guidance of a wheelchair by voice. *In:* IEEE MIDWEST SYMPOSIUM OF CIRCUITS AND SYSTEMS (MWSCAS2000), 1., 2000. *Proceedings…* Michigan; EUA: MWSCAS, 2000. p. 102-107.

KOVÁCS, Z. L. *Redes Neurais artificiais:* fundamentos e aplicações. São Paulo: Acadêmica, 1996.

KROGH, B.; FENG, D. Dynamic steering control of conventionally steered mobile robots. *Journal 0/ Robotic Systems*, v. 4, n. 8, p. 699-721, 1991.

KROGH, B.; GRAETTINGER, T. Evaluation and time-scaling of trajectories for wheeled mobile robots. *Transactions of the ASME*, v. 111, p. 222-231, Jun. 1989.

KRÖHLING, R. A. Algoritmos de controle não convencionais: estudo de caso. 1994. 80f. Dissertação (Mestrado em Engenharia Elétrica) – Universidade Federal do Espírito Santo, 1994.

KWAKEMAAK, H.; SIVAN, R. *Linear optimal control systems*. New York: John Wiley and Sons, 1972.

KYRIAKOPOULOS, K. J.; SARIDIS, G. N. An integrated collision prediction and avoidance scheme for mobile robots in non-stationary environments. *Automatica*, v. 29, n. 2, p. 309-322, March 1993.

LAUMOND, J.-P. Singularities and topological aspects in nonholonomic motion planning. *In:* LAUMOND, J.-P. *Nonholonomic motion planning*. Boston: Kluwer Academic Publishers, 1993. p. 149-199.

LECUN, Y. *Generalisation and network design strategies.* Toronto: Department of Computer Science, University of Toronto, 1989.Technical Report CRG-TR-89-4.

LEONARD, J.; DURRAN-WHYTE, H. Directed sonar sensing/or mobile robot lIQl1igation. Boston: Kluwer Academic Publishers, 1992.

LIN, C. E.; SHEU, Y. A hybrid – control approach for pendulum – car control. *IEEE Transactions on Industrial Electronics*, v. 39, n. 3, p. 208-214, 1992.

LIN, C.-T. Neural Network Based Fuzzy Logic Control and Decision System. *IEEE Transactions on Computers*, v. 40, n. 12, p. 1320-1336, December 1991.

LIPPMAN, R. P. An introduction to computing with neural nets. *IEEE ASSP Magazine*, n. 4, p. 4-22, 1987.

LIU, J.; WU, X.; XIA, J. Visual navigation of a novel economical embedded multi-mode intelligent control system for powered wheelchair. *In:* LIU, J.; WU, X.; XIA, J. *Advances in neural network research and applications*. Berlin: Heidelberg Book, 2010. P. 503-511.

LJUNG, L. *System identification toolboxfor use with Matlabnl*. Natick, Mass.: The MathWorks Inc., 1993.

LJUNG, L. *System identification: theory for the user*. Englewood Cliffs, N. Jersey: Prentice Hall, 1981.

LOMBARDI, A. B.; *Desenvolvimento e modelagem de uma cadeira de rodas servo-assistida para crianças.* 2002. Dissertação (Mestrado em Engenharia Mecânica) - Faculdade de Engenharia Mecanica (FEM), 2002.

LYAPUNOV, A. *The general problem of motion stability* (1907). France: Fac. Sci. Toulouse 9, 1892. p. 203-474.

MACIEJOWSKI, J. M. *Multivariable feedback design*: reading. Mass.: Addinson-Wesley, 1989.

MADEIRA, P. H. A. *Application of human machine interface research in powered wheelchair*. 2008. Dissertação (Mestrado em Engenharia Mecânica), 2008.

MAMDANI, E. H. Application of fuzzy algorithms for the control of a simple dynamic plant. *Proc IEEE*, v. 121, n. 12, p. 121-159, 1974.

MARSDEN, J.; TROMBA, A. *Vector calculus*. USA: W. H. Freeman and Company, 1988.

MÁSSON, E; WANG, Y-J. Introduction to computation and learning in artificial neural networks. *European Journal of Operational Research*, North-Holand, n. 47, p. 1-28, 1990.

MAZO, M. *et al. Wheelchair for physically disabled people with voice, ultrasonic and infrared sensor control*. Autonomous Robots Autonomous Robots, v. 2, n. 3, p. 203-224, 1995.

McCAA, J. Unique wheelchair holds promise for veterans, but lacks support. WFAA. Office of Public and Intergovernmental Affairs. February 15, 2013. Disponível em: <www.bulletinnews.com/va>. Acesso em: 21 set. 2013.

MEDOLA, F. O. *Projeto conceitual e protótipo de uma cadeira de rodas servo-assistida*. 2013. 187f. Tese (Doutorado em Bioengenharia) – Programa de Pós-Graduação Interunidades em Bioengenharia, Universidade de São Paulo, 2013.

MILLER, R. *Engadged*. [S.l.: s.n.], 7 Apr. 2009.

MORAES, H. S. *Projeto conceitual de sistemas de assento para cadeira de rodas:* uma abordagem sistemática. 2009. 143f. Dissertação (Mestrado em Design) - Universidade Federal do Rio Grande do Sul. Escola de Engenharia. Faculdade de Arquitetura. Programa de Pós-Graduação em Design, 2009.

MORATORI, P. B. *Análise de estabilidade e robustez de controladores nebulosos:* aplicação ao controle de trajetória de robôs. 2006. 128f. Dissertação (Mestrado em Informática) – b Programa de Pós-Graduação de Informática, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2006.

MORIN, P.; SAMSON, C. Control of nonlinear chained systems: from the routh-hurwitz stability criterion to time-varying exponential stabilizers. *IEEE Transactions on Automatic Control*, v. 45, n. 1, p. 141-146, 2000.

MUIR, P. F.; NEUMAN, C. P. Kinematic modcliog of wheeled mobile robots. *Journal of Robotic Systems*, v. 4, n. 2, p. 281-340, 1987.MULLER, N.; BEHRINGER, R. Feedforward control for curve steering for an autonomous road vehicle. *Robotics and Automation, IEEE Transactions*, v. 14, n. 5, p. 810-815, May 1992.

MURRAY, R. M. *et al. A mathematical introduction to robotic manipulation*. [*s.l.*]: CRC Press LLC, 1994.

MURRAY, R.; M'CLOSKEY, R. *Exponential stabilization of driftless nonlinear control systems via time-varying, homogeneous feedback*. California: Institute of Technology, Sep. 1994. CDS Tech. Report.

MURRAY, R.; TEEL, A.; WALSH, G. *Nonholonomic control systems:* from steering to stabilization. California: Institutc of Technology, Feb. 1992. CDS Tech. Report.

OLLERO, A.; ULIVI, G.; CUESTA, F. Intelligent control of nonholonomic mobile robots with fuzzy perception. *Fuzzy Sets and Systems*, v. 134, n. 1, p. 47-64, 2003.

ONO, E. *et al.* Vehicle integrated control for steering and traction systems by JI.-synthesis. *Automatica*, v. 30, DO. 11, p. 1639-1647, 1994.

PAPPAS, G. J. ; KYRIAKOPOULOS, K. J. Modeling, stabilization and tracking control of nonholonomic mobile vehicles. *In:* IEEE CONFERENCE, 31., 1992. *Proceedings…* [s.l.], 1992. v. 3. p. 2680-2685.

PEDRYCZ, W.; GOMIDE, F. *An introduction to fuzzy sets*: analysis and design. Cambridge, Massachusetts: The MIT Press, 1998.

PEREIRA, F. L. *et al.* On the design and implementation of a mobile robotic system. *In:* IEEE INTERNATIONAL SYMPOSIUM ON INDUSTRIAL ELECTRONICS, 1997, Slovenia. *Proceedings…* Slovenia: IEEE Service Center, 1997. v. 2.

PEREIRA, F. L.; SOUSA, J. B.; SILVA E. P. A dynamically configurable control architecture for an autonomous underwater vehicle. *In:* INTELLIGENT AUTONOMOUS SYSTEMS, Karlruhe, Germany, Mar. 1995. *Proceedings…* Germany, 1995.

PIRES, G.; NUNES U. A wheelchair steered throuh voice commands and assisted by a reactive fuzzy-logic controller. *Journal of Intelligent and Robotic Systems*, n. 34, p. 301-314, 2002.

PONCE, P.; MOLIN, A.; MENDOZA, R. *Intelligent wheelchair and virtual training* . SIDOROV, G. *et al.* (Eds.). *LabVIEW Advances in Artificial Intelligence.* Linai: MICAI2010, 2010. part 1. p. 422-435.

POPLAWSKI, M.; BIALKO, M. Implementation of fuzzy logic controller in FPGA circuit for guiding electric wheelchair: artificial intelligence and soft computing. *Lecture Notes in Computer Science*, v. 7268, p. 216-222, 2012.

PRADO, M. *et al.* Autonomous mobile robot dynamic constraints due to wheel: ground interaction. *In:* EURISCON 94. May, 1992. *Proceedings…* [*s.l.*] 1992. p. 347-360.

RAMOS, P. *Navegação em robótica móvel baseada em lamdmarkcs naturais.* 1995. Dissertação (Mestrado em Engenharia Elétrica) – Faculdade de Biogenharia da Universidade do Porto, Porto, 1995.

REFENES, A. N. *Stock performance modeling using neural networks:* a comparative study with regression models. London: Department of Computer Sciences, University College London, 1993. p. 1-18.

REFENES, A. N.; AZEMA-BARAC, M.; CHEN, L.; KAROUSSOS, S. A. Currency exchange rate prediction and neural network design strategies. *Neural Computing & Applications*, v. 1, n. 1, p. 46-58, 1993.

RENNO, C. A. *et al.* Neural network-based geometric references recognition applied to ultrasound echo signals. *In:* IEEE MIDWEST SYMPOSIUM, 43., 2000, Michigan, EUA. *Proceedings*... Michigan; EUA: MWSCAS, 2000. v. 3. p. 1344-1347.

RENNO, C. A.; BASTOS FILHO, T. F. Um novo projeto de cadeira robótica para deficientes físicos. *In:* CONGRESO IBEROAMERICANO IBERDISCAP2000, 3., 2000, Madri. *Actas*... (Comunicación Alternativa y Aumentativa y 1o de Tecnologias de Apoyo para la Discapacidad). Madrid, 2000.

RENNO, C. A.; LIMAII, E. J.; PINTO R. L. F. Uma Nova proposta para o desenvolvimento de uma cadeira de rodas robótica equilibrista. *In:* CONEM2012, 4., 2012, São Luís. *Anais*... São Luís, 2012.

RENNÓ, C. A.; LIMAII, E. J.; PINTO R. L. F. A metodology for CG determination. *In:* ISSNIP - IEEE – BIOSIGNALS AND BIOROBOTICS CONFERENCE (BRC), 2013, Rio de Janeiro. *Anais*... Rio de Janeiro, 2013.

RIBEIRO, A. F. *ENIGMA*: cadeira de rodas omnidireccional. SAR, Soluções de Automação e Robótica, Lda, empresa spin-off da Universidade do Minho, 2010.

RIBEIRO, F.; Almeida, L.; Azevedo, J.; Cardeira, C.; Costa, P.; Fonseca, P.; Lima, P. ; Santos V. Mobile Robot Competitions: Fostering Advances In Research, Development And Education In Robotics. Controlo'2000, Guimarães, Portugal, 4-6 October 2000, p. 592-597, ISBN 972-98603-0-0.

RIEDMILLER, M. Advanced supervised learning in multi-layer perceptrons: from backpropagation to adaptative learning algorithms. *Journal of Computer Standards and Interfaces*, v. 16, n. 5, p. 265-278, 1994.

ROSSINI, F. L. *Projeto de controlador robusto aplicado à cadeira de rodas móveis via abordagem por lmis*. 2013. 183f. Dissertação (Mestrado em Engenharia Elétrica) – Universidade Estadual de Londrina, Centro de Tecnologia e Urbanismo, Programa de Pós-Graduação em Engenharia Elétrica, 2013.

RUDZIONIS, V.; MASKELIUNAS, R.; RASYMAS, T. *control of assistive tools using voice interface and fuzzy methods*. Business Information Systems (2012).

RUDZIONIS, V.; MASKELIUNAS, R.; RASYMAS, T. Control of assistive tools using voice interface and fuzzy methods. *Business Information Processing*, v. 117, p. 308-318, 2012.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. *Learning internal representations by error propagation. In:* PARALLEL distributed processing: exploration in the microstructure of cognition. Cambridge: MIT Press, 1986. ICS Report 8506.

SAMSON, C. Path following and time-varying feedback stabilization of a wheeled mobile robot. *International Journal of Robotic Research*, v. 12, n. 1, p. SS-63, Feb. 1992.

SAMSON, C. Time-varying feedback stabilization of car-like wheeled mobile robots. *International Journal of Robotic Research*, v. 12, n. 1, p. SS-63, Feb. 1993.

SANTOS, I. F. *Dinâmica de sistemas mecânicos*. São Paulo: Makron Books, 2001.

SHAW, I. S.; SIMÕES, M. G. *Controle e modelagem fuzzy*. São Paulo: Edgar Blücher, FAPESP, 1999.

SILVA, E. P. *et al.* 00 the design of the PO-ROBOT system. In: 0/ INTELLIGENT VEHICLES SYMPOSIUM, Paris, France, Oct. 1994. *Proceedings…* Paris, 1994.

SILVA, J. F. *Padrões de propulsão para cadeiras de rodas e seus fatores de desempenho*. 2009. Dissertação (Mestrado em Engenharia Mecânica) - Faculdade de Engenharia Mecanica (FEM), UNICAMP, 2009.

SILVA, L. C. de A. *Princípios básicos de um laboratório virtual para veiculos:* aplicação em acessibilidade. 2007. Dissertação (Mestrado em Engenharia Mecânica) – Universidade Estadual de Campinas, Faculdade de Engenharia Mecânica, UNICAMP, Campinas, 2007.

SILVA, P. A. Navegação acústica em ambientes estruturados. 1995. Dissertação (Mestrado em Engenharia Elétrica e Computação) – Faculdade de Engenharia da Universidade do Porto, Porto, Mar. 1995.

SIMPSON R.; LEVINES.Voice control of a powered wheelchair. *IEEE Transaction on Neural System and Rehabilitation Engineering*, v. 10, n. 2, p. 122-125, June 2002.

SIOTINE, J.; LI, W. Applied nonlinear control. Englewood Cliffs, N. Jersey: Prentice Hall, 1991.

SORDALEN, O. J.; CANUDAS, C. Exponential control law for a mobile robot: extension to path following. *IEEE Trans. Robotic and Automation*. v. 9, n. 6, p. 837-842, Dec. 1993.

SOUSA, R. V. Robô agrícola móvel (RAM): uma arquitetura baseada em comportamentos hierárquicos e difusos para sistemas autônomos de guiagem e navegação. 2007. Tese (Doutorado em Engenharia) – Escola de Engenharia de São Carlos, USP, São Carlos, 2007.

ŠPACAPAN, I.; KOCIJAN, J.; BAJD, T. *Simulation of fuzzy-logic-based intelligent wheelchair control system*. *Journal of Intelligent and Robotic Systems*, v. 39, n. 2, p. 227-241, 2004.

SRINIVASAVARADHAN, L.; CHANDRAMOULI, G. Automated vehicles for physically and visually challenged. *In:* MEMSTECH’2008, May 21-24, Polyana, UKRAINE, 2008. *Proceedings…* Ukraine, 2008.

STEER, B. Trajectory planning for a mobile robot. *International Journal 0/ Robotic Research*, v. 8, n. S, p. 1-14, Oct. 1989.

STEFANOV, D.; AVTANSKI, A.; BIEN, Z. Z. *a concept for control of indoor-operated autonomous wheelchair*. Advances in Rehabilitation Robotics (2004).

STEVENS, JOHN K. Reverse engineering the brain. *BYTE*, p. 287-289, April 1985.

SUGIE, H. *et al.* Toward human-friendly robot systems. *IEEE Robotics and Automation*, v. 2, p. 2181-2186, 1995.

SURKAN, A. J; SINGLETON, C. *Neural networks for bond rating improved by multiple hidden layers*. *In:* IEEE INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS, 1990, San Diego. *Proceedings…* San Diego: [*s. n.*], 1990. p. 163-168.

SZWARCFITER, J. L.; MARKENZON, L. *Estruturas de dados e seus algoritmos***.** 2. ed. rev. Rio de Janeiro: LTC, 1994. cap. 1.

TAKAHASHI, Y. *et al.* Back and forward moving scheme of front wheel raising for inverse pendulum control wheel chair robot. *In:* IEEE INTERNATIONAL CONFERENCE ON ROBOTICS 8 AUTOMATION, Seoul, Korea. May 21-26, 2001. Proceedings… Korea, 2001.

TURENNOUT, P. *et al.* Wall-following control of a mobile robot. *In:* CONF. ROBOTICS AND AUTOMATION, 1992, Nice, France. *Proceedings…* Nice, May 1992. p. 280-285.

UNEMI, T. *et al.* Toward human-friendly robot systems. 1995 IEEE. *Laboratory* for International Fuzzy Engineering Research Siber Hegner Building 4FL. 89-1 Yamashitacho. Naka-ku, Yokohama 231, Japan.

VIDYASAGAR, M. *Nonlinear systems analysis*. Englewood Cliffs, N. Jersey: Prentice Hall,1978.

WALSH, G. *et al.* Stabilization of trajectories for systems with nonholonomic constraints. *Automatic Control, IEEE Transactions*, v. 39, n. 1, p. 216-222, 1994.

WASSERMAN, P. D. *Neural computing:* theory and practice. New York: Van Nostrand Reinhold, 1989.

WEN, J. T.-Y.; KREUTZ, K.-D. The attitude control problem. *IEEE Transactions on Automatic Control*, v. 36, p. 1148-1162, October 1991.

XUEEN, L.; TIENIU, T.; XIAOJIAN, Z. *Multi-modal navigation for interactive wheelchair*. Advances in Multimodal Interfaces — ICMI 2000 (2000).

YAMAKAWA, T. Stabilization of an inverted pendulum by a high-speed fuzzy logic controller hardware system. *Fuzzy Sets and Systems*, n. 32, p. 161-180, 1987.

YASUNOBU, S.; MIYAMOTO, S.; IHARA, H. A fuzzy control for train automatic stop control. *Trans. of the Society ofInstrument and Control Engineers*, v. E-2, n.1, p. 1-9, 2002.

YOUNGOHC, Y. *et al.* Comparison of discriminant analysis vs artificial neural networks. *Journal of the Operational Research Society*, n. 44, p. 52-55, 1991.

ZADEH, L. A. Fuzzy sets and systems. *In:* FOX, J. (Ed.). *System theory*. Brooklyn, NY: Polytechnic Press, 1965b. p. 29-39.

ZADEH, L. A. Fuzzy sets. *Information and Control*. n. 8, p. 338-353, 1965a.

ZHAO, Y.; BEMENT, S. L. Kinematics, dynamics and control of wheeled mobile robots. *In:* CONF. ROBOTICS AND AUTOMATION, 1992, Nice, France. *Proceedings…* Nice, May 1992. P. 91-96

ZHU, S. Q.; LEWIS, F. L. Maneuver planning and robust path tracking for mobile robot nonholonomic systems. *In:* IEEE MEDITERRANEAN SYMPOSIUM ON NEW DIRECTIONS IN CONTROL &: AUTOMATION, 2., Crete. Greece, Jun. 1994. *Proceedings…* Crete, 1994. P. 529-539.

XIAO, J. *et al.* Fuzzy controller for wall-climbing microrobots. Fuzzy Systems, IEEE Transactions, v. 12, n. 4, p. 466-480, 2004.

APÊNDICES

APD.1. Histórico e Fundamentação Teórica

Uma **cadeira de rodas** é uma cadeira montada sobre rodas que é utilizada por indivíduos com dificuldade de locomoção, normalmente chamados de cadeirantes, podendo ser empurrada por alguém, movida manualmente ou eletronicamente pelo ocupante. Por ser largamente utilizada por pessoas portadoras de variados graus e tipos de deficiência física, é também utilizada no símbolo que indica acesso a pessoas com estas necessidades especiais. Depois de fabricadas, muitas cadeiras de rodas passam por médicos/fisioterapeutas para serem adaptadas ao portador daquele tipo de deficiência física.

A cadeira de rodas é um objeto indispensável para pessoas que apresentam dificuldade de locomoção, podendo-se encontrar representação de seu uso em alguns artefatos de IV AC na Antiga Grécia. Com o aprimoramento de sua fabricação ao longo do tempo, obtém-se diversos modelos para atender as diferentes necessidades de seus usuários. Acredita-se que os egípcios foram os primeiros a utilizarem a cadeira de rodas, como uma espécie de carrinho de mão para transportar pessoas, entretanto em alguns objetos gregos há gravuras feitas em torno do século IV AC de Hefesto, deus grego da metalurgia, retratando a utilização da cadeira de rodas. Responsável por ser o ferreiro dos deuses, ele era adorado por artesões, metalúrgicos e era também conhecido entre os romanos politeístas por ser o deus dos vulcões. Nitidamente na ilustração, observa-se Hefesto acomodado sobre uma cadeira de rodas com aros e dois cisnes para movimentá-la, dando alusão de ser autopropulsora e utilizável tanto em água como em terra, ou seja, não necessitava da força do ocupante. Outra representação encontrada na abertura de um vaso grego observa-se novamente, o deus grego Hefesto como na primeira ilustração sobre uma cadeira de rodas, sendo recebido como um conviva entre os demais deuses do Olimpo. Observa-se que na cultura grega, em outro artefato raro encontrado, uma pintura representando uma cama adaptada com rodas, possivelmente usada como berço, do século VI AC.

Nos próximos séculos, destacou-se a cadeira usada pelo Rei Filipe II da Espanha, em 1595, em plena era das navegações. Elaborou-se naquela época uma cadeira de rodas que atendesse a todas as necessidades do monarca, como engenhos de inclinação, descanso para os pés e que modifica-se para leito temporário de repouso, pois o rei necessitava de muita precaução para ser locomovido e ter acesso aos diversos recintos do castelo.

Com o passar dos séculos muitas famílias providas de riquezas, encomendavam cadeiras para seus membros, pois não existia a fabricação ordenada de cadeira de rodas. Então se deu a fabricação dessa similar poltrona, feita com vime da Índia, assento, duas rodas traseiras grandes e duas rodas pequenas na frente para a movimentação ser garantida, pesando em torno de 25 kg.

No ano de 1655, Stephan Farfler, um relojoeiro paraplégico, aos 22 anos de idade criou um modelo triciclo. Além de ser confortável, era movimentado pelo próprio usuário. Utilizava os dois braços e não requeria qualquer ajuda em terreno plano - desde que não houvesse barreiras, como hoje em dia.

No ano de 1933 Herbert A. Everest, norte-americano, encomendou uma cadeira de rodas que poderia ser levada em um automóvel, que foi um passo decisivo para o objetivo desenvolvimento de cadeiras de rodas mais versáteis. O engenheiro H.C. Jennings construiu para ele a primeira cadeira de rodas dobrável. Esse modelo, devidamente patenteado nos Estados Unidos como muitos outros modelos, foi utilizado por décadas, com a marca Everest/Jennings, antes que outros surgissem no mercado.

As cadeiras de rodas evoluíram de uma forma surpreendente desde as primeiras décadas do Século XX graças ao avanço industrial e com o surgimento de matéria-prima muito mais

moldável e mais leve, como o alumínio e aços-liga, provocaram em muito uma maior demanda. Seria tarefa impossível levantar todos os modelos existentes, desde as manuais, dobráveis ou não, às hospitalares, às adaptadas a situações específicas e também às motorizadas e por que não dizer robóticas, que aos poucos vão tomando conta do mercado.

Há modelos para muitos gostos e muitas necessidades. De triciclos (scooters) existem centenas de modelos, cores e estilos em todas as partes do mundo. Também há cadeiras de rodas para todos os terrenos e superação de obstáculos. O importante é que elas não sejam confinadoras, mas libertadoras das pessoas que as utilizam.

Uma das figuras mais famosas e importantes que usava cadeira de rodas, foi Franklin Delano Roosevelt. Ele viveu no Século XX e foi vítima da poliomielite quando já era Presidente dos Estados Unidos, quando engajou toda a nação americana na Segunda Guerra Mundial. Embora fosse notório que evitava ser visto em público ou fotografado utilizando uma cadeira de rodas, não a dispensava em momentos especiais. Dava seu apoio à campanha nacional contra a poliomielite.

APD.1.1. Custo computacional

Sabemos que um algoritmo é um conjunto de instruções que definem passos para a resolução de um problema. O estudo de algoritmos envolve dois aspectos básicos: a correção e a eficiência. A correção de um algoritmo refere-se à exatidão da solução que ele oferece para um determinado problema, ou seja, analisa se o algoritmo resolve o problema sem erros. Por outro lado, a eficiência de um algoritmo considera se a solução que ele propõe utiliza os recursos computacionais, especialmente memória e tempo de processamento, de maneira adequada e econômica.

Uma das formas mais simples de avaliar a eficiência de um algoritmo é através da análise empírica, isto é, por meio da experimentação e observação da execução do programa correspondente ao algoritmo. Esse tipo de avaliação enfrenta grandes desafios, pois se baseia no desenvolvimento de uma implementação correta e completa do algoritmo e depende não apenas da natureza dos dados de entrada utilizados durante a experimentação, mas também de outros fatores que têm influência no experimento. Por exemplo, ao comparar algoritmos empiricamente é necessário levar em conta o ambiente de execução utilizado, considerando as máquinas, os compiladores e os sistemas utilizados durante os experimentos. Portanto, um dos possíveis problemas da análise empírica é que uma implementação pode ter sido realizada com mais cuidado (*otimizada*) do que a outra, levando a comparações equivocadas.

A *análise matemática de algoritmos* é uma alternativa à análise empírica, sendo especialmente útil quando a análise experimental consome uma quantidade significativa de tempo ou quando necessitamos de alguma indicação de eficiência antes de qualquer investimento de desenvolvimento. Nesta aula, você vai aprender noções sobre análise matemática da eficiência de algoritmos.

Análise matemática de algoritmos

A qualidade de um programa de computador depende da eficiência de seu algoritmo para resolver o problema ao qual ele propõe solução. Do ponto de vista do usuário de um programa, a qualidade pode ser medida em termos de diversos fatores, como interface, robustez, compatibilidade, desempenho, consumo de recursos, entre outros. Do ponto de vista

do programador, a qualidade de um programa pode estar relacionada a critérios como portabilidade, clareza e reuso. Por outro lado, a análise matemática de algoritmos baseia-se em critérios mais gerais para medir a qualidade de um algoritmo.

O custo exigido pela maioria dos algoritmos geralmente pode ser definido matematicamente em função de seu principal dado de entrada, ao qual chamaremos de *parâmetro primário* do algoritmo. Faremos considerações sobre o tamanho N do parâmetro primário, onde esse tamanho corresponde ao número de elementos que compõem o parâmetro primário. Por exemplo, se o parâmetro primário de um algoritmo for um vetor de inteiros, N representaria o número de elementos desse vetor. Portanto, o tamanho do parâmetro primário afeta diretamente o custo de execução do algoritmo, sendo diretamente proporcional a ele. O objetivo da análise matemática é expressar a necessidade de recursos de um programa (tempo de execução e/ou espaço de memória) em termos desse parâmetro primário N.

Quando comparada à análise empírica, que é uma análise baseada na experimentação e na observação da execução do programa que implementa um algoritmo, a análise matemática de algoritmos faz uma avaliação mais informativa e mais barata, pois é uma análise independente da máquina, compilador ou sistema e não exige a criação e execução de um programa correspondente.

Crescimento de funções

As expressões produzidas pela análise matemática da eficiência de um algoritmo geralmente são baseadas em funções matemáticas bem conhecidas. Uma das vantagens de associar a eficiência de um algoritmo a essas funções é que fica mais fácil entender como seu custo varia em relação ao tamanho do parâmetro primário do algoritmo. Outra vantagem é que isso nos permite comparar os custos de diferentes algoritmos com facilidade. A seguir, você poderá ver as funções matemáticas normalmente utilizadas na análise matemática do custo:

1: maior parte das instruções é executada apenas uma ou um número constante de vezes, independente do parâmetro primário N (tempo de execução ***constante***);

$logN$: ocorre em algoritmos que resolvem um problema maior transformando-o em uma série de subproblemas, reduzindo o tamanho do problema por certa constante fracionária a cada passo tempo de execução ***logarítmico***). Sempre que N dobra, a complexidade do algoritmo aumenta por uma constante, mas não dobra até que o tamanho da entrada seja N^2;

N: acontece quando uma pequena quantidade de processamento deve ser feito para cada elemento da entrada (tempo de execução ***linear***). Corresponde ao menor custo possível para um algoritmo que deve processar N entradas ou gerar N saídas;

$N.logN$: ocorre em algoritmos que quebram o problema principal em subproblemas menores, resolvendo-os e depois combinando as soluções (tempo de execução ***linearítmico*** ou N log N).Quando N dobra, o tempo de execução do algoritmo aumenta um pouco mais do que o dobro;

N^2: complexidade de algoritmos que processam todos os pares de itens de dados (tempo de execução ***quadrático***). Esse tipo de complexidade é aceitável apenas para problemas relativamente pequenos, pois quando N dobra, o tempo de execução aumenta 4 vezes;

N^3: complexidade de algoritmos que processam todas as combinações de itens de dados três a três (tempo de execução ***cúbico***). Quando N dobra, o tempo de execução aumenta 8 vezes.

2^N: corresponde a algoritmos que utilizam força bruta na solução de problemas (tempo de execução ***exponencial***). Algoritmos com esse desempenho são impraticáveis para problemas práticos, pois sempre que N dobra o tempo de execução do algoritmo é elevado ao quadrado.

A Figura A1.1 mostra um gráfico onde você pode analisar como as curvas de crescimento correspondentes a essas funções se comportam à medida que aumentamos o tamanho do parâmetro primário do algoritmo. Neste gráfico, a curva da função de custo f é rotulada como $O(f)$, antecipando a notação que você conhecerá daqui a pouco, ainda nesta aula. Para a maioria dessas funções, quanto maior for o tamanho do parâmetro primário N, maior será o custo computacional do algoritmo. Dessa forma, você poder ver, por exemplo, que se um algoritmo tem sua expressão de custo correspondente a uma função constante, então, ele consome sempre o mesmo tempo, independente do valor de N. Por outro lado, se o algoritmo tem uma expressão de custo exponencial, um pequeno aumento no tamanho de N tem grande impacto no tempo consumido pelo algoritmo.

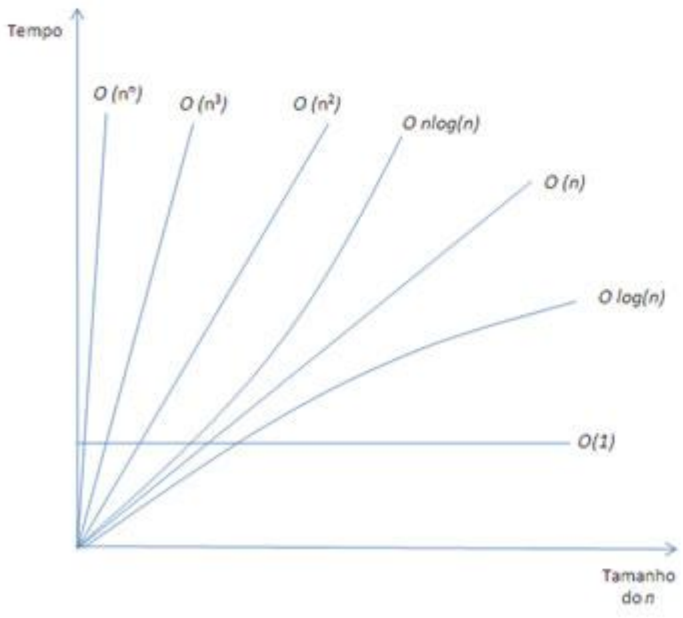


Figura A1.1 – Funções de custo tradicionais.

O custo de um algoritmo normalmente é descrito por uma expressão formada por alguma dessas funções multiplicada por uma constante. Esta expressão corresponde à operação dominante, obtida após desconsiderar outros termos de menor importância, que correspondem a operações de menor impacto no custo do algoritmo. Por exemplo, a expressão $c_1.N^2 + c_2.N + c_3$ poderia ser utilizada para denotar o custo de um algoritmo, onde N seria o parâmetro primário e c_1, c_2 e c_3 seriam as constantes multiplicativas que representam tempos de execução de operações elementares. Dada a expressão de custo de um algoritmo, o termo de maior importância para a determinação do custo corresponde à operação dominante, geralmente relacionada ao número de comandos da instrução de repetição mais interna do algoritmo. Normalmente, representamos o custo do algoritmo de forma simplificada, considerando apenas o termo de maior importância na expressão de custo global. Assim, a

expressão de custo anterior poderia ser simplificada para N^2, que é o termo que mais influencia no custo à media que o tamanho do parâmetro primário aumenta.

As constantes de uma expressão de custo representam os tempos de execução de operações elementares (atribuição, leitura e escrita simples etc.). Esses tempos normalmente não podem ser determinados apenas pela observação do algoritmo, pois vão depender do ambiente de execução (velocidade do processador, velocidade de acesso memória do computador etc.).

Simplificações

As simplificações das expressões matemáticas são necessárias para facilitar o processo de análise de custos e comparação entre algoritmos. A seguir, mostramos as principais simplificações aplicadas na análise matemática do custo de um algoritmo:

considere a execução do algoritmo quando aplicado a quantidades de dados *suficientemente* grandes. Em outras palavras, assuma que o parâmetro primário N tende a infinito;

ignore *constantes* aditivas ou multiplicativas na fórmula matemática correspondente ao custo. Essas constantes não interferem no comportamento geral da função matemática associada;

analise o algoritmo dividindo-o em etapas ou passos elementares, sendo que cada passo envolve um número fixo de *operações básicas* cujos tempos de execução são assumidos *constantes*. Nesse contexto, o que chamamos de operações básicas são as operações muito simples e que não podem ser subdivididas em outros passos, como uma atribuição ou um teste lógico;

o número de passos do algoritmo é dado pelo número de passos da operação básica de maior frequência, comumente chamada de *operação dominante*.

O custo de um algoritmo normalmente é medido em termos de tempo de processamento (*complexidade temporal*) ou consumo de memória (*complexidade espacial*). A complexidade temporal é uma medida de quanto tempo o algoritmo leva para executar, enquanto a complexidade espacial é uma medida de quanto espaço de memória o algoritmo precisa para executar até o fim. Na análise matemática, ambas as complexidades são determinadas com base no tamanho do parâmetro primário do algoritmo. Daqui em diante, consideraremos o custo do algoritmo sendo medido com base em seu tempo de execução, portanto, nos concentraremos sobre a complexidade temporal.

A seguir, você verá alguns exemplos de algoritmos e suas complexidades temporais. Não se esqueça de observar que o custo temporal é dado em termos das funções matemáticas apresentadas anteriormente, e que aplicamos as regras de simplificação da expressão de custo naturalmente, por exemplo, ao escolher o termo mais importante da expressão de custo para representar a complexidade global do algoritmo.

Exemplo 1

Algoritmo 1: Complexidade Constante

Como exemplo de algoritmo com complexidade constante, considere o algoritmo abaixo, utilizado para acessar um elemento em um vetor:

```
1  programa
2      var i, n: inteiro
3      var v: arranjo de n inteiro
4
5      leia i                                               #c1
6      leia n                                               #c2
7      leia v                                               #c3
8      se i > n então                                       #c4
9          escreva ("Acesso fora dos limites do vetor!")    #c5
10     senão
11         escreva v[i]                                     #c6
12     fim
13 fim
```

Nesse algoritmo e nos outros mostrados nesta aula, associamos custos às principais instruções, identificados sob a forma de comentários. Nesse exemplo, o tempo de execução T depende dos valores de i e n, onde n corresponde ao parâmetro primário do algoritmo. Se $i > n$ então, T deverá ser calculado levando em conta o tempo gasto em uma comparação (linha 8) mais o tempo gasto para escrever uma mensagem informando o erro (linha 9); se $i \leq n$ então, T deverá ser calculado levando em conta o tempo gasto em uma comparação (linha 8) mais o tempo gasto em um acesso ao vetor v (linha 11). Dessa forma, o tempo de execução T desse algoritmo será dado por $c_1 + c_2 + c_3 + c_4 + Máximo(c_5, c_6)$, onde $Máximo(c_5, c_6)$ representa o valor máximo entre c_5 e c_6. Note que o tempo de execução é limitado por um número fixo de operações com custos constantes, independente do tamanho da entrada, isto é, independente do tamanho n do vetor v. Se trocarmos todas as constantes anteriores por uma única constante k que represente o valor correspondente à expressão $c_1 + c_2 + c_3 + c_4 + Máximo(c_5, c_6)$, a função de custo reduz a esta constante k. , a complexidade desse algoritmo é dita **constante** (ou seja, **1**).

Exemplo 2

Algoritmo 2: Complexidade Linear

Como exemplo de algoritmo com complexidade linear, considere o algoritmo a seguir, responsável por achar o máximo elemento de um vetor. Neste algoritmo, o tamanho n do vetor v corresponde ao parâmetro primário.

```
programa
|   var n, i, max: inteiro
|   var v: arranjo de n inteiro
|
|   leia n                                                  #c1
|   leia v                                                  #c2
|   se n = 0 então                                          #c3
|   |       escreva ("Máximo chamado com vetor vazio")      #c4
|   senão
|   |       max ← v[1]                                      #c5
|   |       para i de 1 até n faça                          #c6
|   |       |   se v[i] > max então                         #c7
|   |       |   |       max ← v[i]                          #c8
|   |       |   fim
|   |       fim
|   fim
|   escreva max                                             #c9
fim
```

Considerando as constantes identificadas nos comentários acima, o tempo T execução será dado por $c_1 + c_2 + c_3 + Máximo(c_4, c_5 + n.(c_6 + c_7 + c_8)) + c_9$, onde $Máximo$ é a função definida anteriormente. Como o termo de maior importância do custo é determinando quando consideramos valores grandes para o parâmetro primário n, podemos simplificar a expressão de custo desse algoritmo para $k_1 + c_5 + n.k_2 + c_9$, onde $k_1 = c_1 + c_2 + c_3$ e $k_2 = c_6 + c_7 + c_8$. Eliminando-se as constantes aditivas (de acordo com a regra apresentada anteriormente), temos $T \leq k_2 \cdot n$, onde k_2 é uma constante que depende das características do sistema onde o algoritmo será executado. Portanto, a complexidade máxima deste algoritmo é **linear** (ou seja, n).

Exemplo 3

Algoritmo 3: Complexidade Quadrática

Considere o algoritmo abaixo, utilizado para transpor uma matriz quadrada m. Os principais dados do algoritmo são a matriz m e seu tamanho n, sendo que esse corresponde a seu parâmetro primário.

```
1   programa
2       var n, i, j, aux: inteiro
3       var m: arranjo de n x n inteiro
4
5       leia n
6       leia m
7       para i de 1 até n-1 faça                  #c1
11          para j de i+1 até n faça              # c2
12              aux ← m[i, j]                     #c3
13              m[i, j] ← m[j, i]                 #c4
14              m[j, i] ← aux                     #c5
15          fim
16      fim
18  fim
```

A partir de agora, apenas as instruções que têm maior influência sobre o custo global do algoritmo terão seu custos individuais identificados. Essa decisão não afetará o cálculo do custo global do algoritmo, uma vez que apenas o termo mais significativo é considerado na expressão final do custo. Nesse algoritmo, o *para* externo (linha 07) é executado $n-1$ vezes, enquanto a complexidade total do *para* interno (linha 11) é dado por uma expressão L. Assim, a complexidade global do algoritmo é $T = (n-1).(c_1 + L)$, onde $L = (n-j).(c_2 + c_3 + c_4 + c_5)$. Desenvolvendo essas expressões, teremos:

$$
\begin{aligned}
T &= n.c_1 + n.L - 1.c_1 - 1.L \\
&= n.(c_1 + L) - c_1 - L
\end{aligned}
$$

Por outro lado, sabemos que a cada iteração do *para* interno a seguinte expressão é verdadeira:

$$
\begin{aligned}
L &\leq (n-2).(c_2 + c_3 + c_4 + c_5) \\
&\leq n.k_1 + k_2
\end{aligned}
$$

onde $k_1 = c_2 + c_3 + c_4 + c_5$ e $k_2 = -2.(c_2 + c_3 + c_4 + c_5)$. Portanto,

$$
\begin{aligned}
T &\leq n.(c_1 + n.k_1 + k_2) - c_1 - n.k_1 - k_2 \\
&\leq n.c_1 + n^2.k_1 + n.k_2 - c_1 - n.k_1 - k_2 \\
&\leq n^2.k_1 + n.(c_1 + k_2 - k_1) + (-c_1 - k_2)
\end{aligned}
$$

Logo, $T \leq k_1.n^2 + k_3.n + k_4$, onde $k_3 = c_1 + k_2 - k_1$ e $k_4 = -c_1 - k_2$, e o algoritmo tem **complexidade quadrática** (ou seja, n^2).

Apesar de diferentes, os dois algoritmos seguintes realizam o mesmo cálculo: a soma dos elementos da diagonal principal de uma matriz quadrada de ordem n. Qual a complexidade de cada algoritmo?

a) Algoritmo 1

```
programa
    var i, soma: inteiro
    var m: arranjo de n x n inteiro

    leia m
    soma ← 0
    para i de 1 até n faça
        soma ← soma + m[i, i]
    fim
    escreva ("A soma dos elementos da diagonal principal é: " + soma)
fim
```

b) Algoritmo 2

```
programa
    var i, j, soma: inteiro
    var m: arranjo de n x n inteiro

    leia m
    soma ← 0
    para i de 1 até n faça
        para j de 1 até n faça
            se i = j então
                soma ← soma + m[i, j]
            fim
        fim
    fim
    escreva ("A soma dos elementos da diagonal principal é: " + soma)
fim
```

Notações de complexidade

Neste momento, aprofundaremos um pouco mais nossa discussão acerca da complexidade de algoritmos. Até o momento, aprendemos a medir a complexidade de um algoritmo, observadas as devidas simplificações. Agora, conheceremos a notação mais empregada para expressar e comparar complexidades algorítmicas. Para tanto, conheceremos as complexidades de *pior caso*, *melhor caso* e *caso médio*. Nossa discussão seguinte será concentrada na notação de complexidade de pior caso.

Complexidade de pior caso, melhor caso e caso médio

Em alguns casos, faz-se necessário analisar o comportamento de um mesmo algoritmo considerando diferentes entradas de dados e seus impactos sobre o custo do algoritmo. Seja A um algoritmo e $\{E_1,\ldots,E_m\}$ o conjunto de todas as entradas possíveis de A. Entenda por t_i o número de passos efetuados por A quando a entrada for E_i. Definimos:

complexidade do **pior caso**: t_i, onde t_i o **maior** número de passos efetuado por A, correspondente à entrada E_i;

complexidade do **melhor caso**: t_i, onde t_i o **menor** número de passos efetuado por A, correspondente à entrada E_i;

complexidade do **caso médio**: $\sum_{i=1}^{m} p_i . t_i$, onde p_i é a probabilidade de ocorrência da entrada E_i;

Para entender melhor o significado desses casos, considere que o algoritmo A tem por função ordenar os elementos de um vetor com n inteiros, realizando sucessivas operações sobre esses elementos na medida em que o vetor é ordenado. Sendo o custo desse algoritmo determinado pelo número de operações realizadas, se essas operações forem realizadas apenas para os elementos desordenados, o melhor caso desse algoritmo será quando o vetor fornecido na entrada já estiver ordenado. O pior caso será aquele em que o vetor fornecido na entrada tem seus elementos dispostos na ordem inversa à desejada.

A complexidade do pior caso considera o número de passos que o algoritmo efetua para a entrada mais desfavorável, para a qual é necessário executar o maior número de instruções. Esse é o caso onde os dados de entrada fazem com que o algoritmo consuma mais tempo do que para qualquer outra entrada. A complexidade de pior caso é uma das análises mais importantes porque fornece um teto para o número de passos que o algoritmo pode efetuar, independente da entrada considerada. Em outras palavras, se você conhecer a complexidade de pior caso de um algoritmo, saberá que seu custo nunca irá superar um determinado limite, tendo assim uma garantia acerca do custo máximo do algoritmo.

Limite superior

A notação mais comumente usada para medir e comparar algoritmos é O (big-Oh), útil para estabelecer uma ordem relativa entre funções, permitindo comparar suas taxas de crescimento. Essa notação é a geralmente utilizada para definir a complexidade de pior caso, expressando um limite superior do custo de um algoritmo. Essa notação compara o custo $g(n)$ de um algoritmo a um custo $f(n)$, onde n corresponde ao parâmetro primário do algoritmo, informando que o custo $g(n)$ nunca é superior ao custo $f(n)$. Portanto, $f(n)$ funciona como um limite superior para o custo $g(n)$.

Formalmente, dizemos que a função de custo $g(n)$ é $O(f(n))$ ou, simplesmente, $g = O(f)$, se existir uma constante $c_0 > 0$ e um inteiro n_0, tais que $g(n) \leq c_0 . f(n)$ quando $n > n_0$ onde f e g são funções reais positivas de variável inteira n. A Figura A1.2 mostra graficamente o comportamento das funções g e f onde o eixo das abscissas corresponde ao tamanho do parâmetro primário e o eixo das ordenadas corresponde ao valor de custo dessas funções.

e

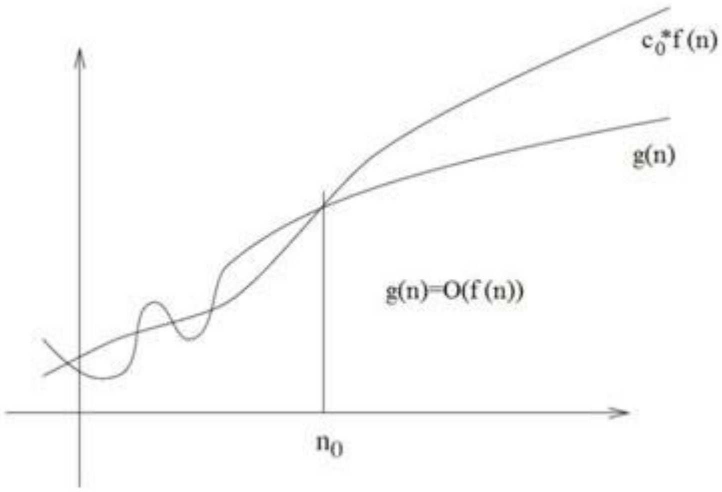


Figura A1.2 – Limite superior de complexidade

Você está se perguntando o que significam os outros símbolos envolvidos nesse gráfico, não é mesmo? O gráfico nos diz que se o tamanho do parâmetro primário n for maior que n_0 (esse é o significado da expressão $n > n_0$, o custo $g(n)$ nunca será maior que o custo $c_0.f(n)$ (esse é o significado da expressão $g(n) \leq c_0.f(n)$), onde a constante estritamente positiva c_0 funciona como um fator de ajuste para $f(n)$ (o que importa é a taxa de crescimento de f, por isso usamos esse fator de ajuste). A partir de n_0, a função g fica sempre *menor ou igual* à função f. Portanto, g não cresce mais rapidamente que f e dizemos que $g(n)$ é $O(f(n))$.

Como exemplo, considere que as expressões de custo de dois algoritmos foram medidas como $g(n) = 1000.n$ e $f(n) = n^2$, onde n corresponde ao parâmetro primário comum a esses algoritmos. Podemos dizer que $g(n)$ é $O(f(n))$, porque se o tamanho do parâmetro primário n for superior a $n_0 = 999$, temos que $g(n) \leq c_0.f(n)$ para $c_0 = 1$. Portanto, $g(n) = O(f(n))$ para $n_0 = 999$ e $c_0 = 1$.

Expressões do tipo $O(f(n))$ são chamadas de expressões *assintóticas*. A terminologia *expressão assintótica* é comumente utilizada no contexto de análise de complexidade de algoritmos. Se o custo é dado por uma expressão assintótica, então, o esforço do algoritmo foi medido considerando seu comportamento diante de um tamanho extremamente grande do parâmetro primário. Na verdade, a análise matemática é feita assumindo que o tamanho do parâmetro primário tende a infinito, ou seja, analisa o algoritmo para grandes entradas de dados. Dessa forma, a análise considera a execução do algoritmo em condições extremas, processando grades quantidades de dados. No caso da complexidade de pior caso, além da grande quantidade de dados assumida, consideramos, também, que os dados da entrada estão dispostos da maneira mais desfavorável possível para o algoritmo.

Existem também notações para *análise de limite inferior* (notação Ω – big Omega) e para a *análise de limite exato* (notação Θ – big Theta). Contudo, não discutiremos essas outras formas de análise, pois a análise do pior caso é a notação mais utilizada e normalmente a mais fácil de ser calculada.

Comparação de algoritmos

Para se *comparar* dois algoritmos, devemos primeiramente calcular suas complexidades assintóticas. Dadas as *expressões assintóticas* de dois algoritmos diferentes, f e g , podemos compará-los da seguinte forma:

se f é sempre inferior a g , ou seja, o gráfico de f está sempre abaixo do gráfico de g , então, escolhemos o algoritmo correspondente a f ;

se f às vezes é inferior a g , e vice-versa, e os gráficos de f e g , têm um número infinito de interseções, então, temos um empate, ou seja, a função de custo *não* é suficiente para escolher entre os dois algoritmos;

se f às vezes é inferior a g , e vice-versa, e os gráficos de f e g , têm um número finito de pontos de interseção, então, a partir de certo valor n, f é sempre superior a g , ou é sempre inferior. Neste caso, escolhemos o algoritmo cuja função é inferior para grandes valores de n.

Em algumas estruturas de dados, a complexidade de uma operação pode depender do estado da estrutura. Por exemplo, em determinados momentos, uma inserção pode ter um custo constante, em outros momentos, um custo linear. Nesse caso, aplica-se a complexidade amortizada, que permite expressar a complexidade média da operação, considerando todos os estados possíveis. Essa notação foge ao escopo de nossa discussão atual.

APD.1.2. Inteligência Artificial

Inteligência do Latim *intellectus*, de *intelligere* = inteligir, entender, compreender. Composto de *íntus* = dentro e *lègere* = recolher, escolher, ler (cfr. *intendere*) pode ser definida como a capacidade mental de raciocinar, planejar, resolver problemas, abstrair ideias, compreender ideias, linguagens e aprender. Embora pessoas leigas geralmente percebam o conceito de inteligência sob um âmbito maior, na Psicologia, o estudo da inteligência geralmente entende que este conceito não compreende a criatividade, o caráter ou a sabedoria. Conforme a definição que se tome, pode ser considerado um dos aspectos da personalidade.

Existem dois "consensos" na definição de inteligência. O primeiro,de *Intelligence: Knowns and Unknowns*, um relatório de uma equipe congregada pela Associação Americana de Psicologia , em 1995:

"*Os indivíduos diferem na habilidade de entender idéias complexas, de se adaptarem com eficácia ao ambiente, de aprenderem com a experiência, de se engajarem nas várias formas de raciocínio, de superarem obstáculos mediante o pensamento. Embora tais diferenças individuais possam ser substanciais, nunca são completamente consistentes: o desempenho intelectual de uma dada pessoa vai variar em ocasiões distintas, em domínios distintos, a se julgar por critérios distintos. Os conceitos de inteligência são tentativas de aclarar e organizar esse conjunto complexo de fenômenos.*"

Uma segunda definição de inteligência vem de *Mainstream Science on Intelligence*, que foi assinada por cinquenta e dois pesquisadores em inteligência, em 1994:

“*uma capacidade mental bastante geral que, entre outras coisas, envolve a habilidade de raciocinar, planejar, resolver problemas, pensar de forma abstrata, compreender ideias complexas, aprender rápido e aprender com a experiência. Não é uma mera aprendizagem literária, uma habilidade estritamente acadêmica ou um talento para sair-se bem em provas. Ao contrário disso, o conceito refere-se a uma capacidade mais ampla e mais profunda de compreensão do mundo à sua volta - 'pegar no ar', 'pegar' o sentido das coisas ou 'perceber' uma coisa.*”

Adicionalmente segundo Herrnstein e Murray: "... está associada a habilidade cognitiva" e segundo Sternberg e Salter: "... ao comportamento adaptativo orientado a metas". Além do que definiu Saulo Vallory: "...habilidade de intencionalmente reorganizar informações para inferir novos conhecimentos". De acordo com a Teoria da modificabilidade cognitiva estrutural, do psicopedagogo Reuven Feuerstein, todo ser humano com dificuldades de aprendizado, em qualquer fase de sua vida, pode ter sua inteligência "amplificada". Isto, daria a qualquer indivíduo a capacidade de aprender.

Nas propostas de alguns investigadores, a inteligência não é uma só, mas consiste num conjunto de capacidades relativamente independentes. O psicólogo Howard Gardner desenvolveu a teoria das inteligências múltiplas, identificando sete diferentes tipos inteligência: lógico-matemática, linguística, espacial, musical, cinemática, intrapessoal e interpessoal. Mais recentemente, Gardner expandiu seu conceito acrescentando à lista a inteligência naturalista e a inteligência existencial. [fonte:Wikipédia]

Diferentemente do que se acreditava no início da cultura humana que dizia que a inteligência residia no coração (a emoção?), sabe-se que seu fiel depositário é o cérebro humano onde reside toda a ansiedade, comportamentos, idéias e concepções. Este maravilhoso orgão abriga toda a nossa criatividade e segredos ainda bem guardados que parecem infinitos.

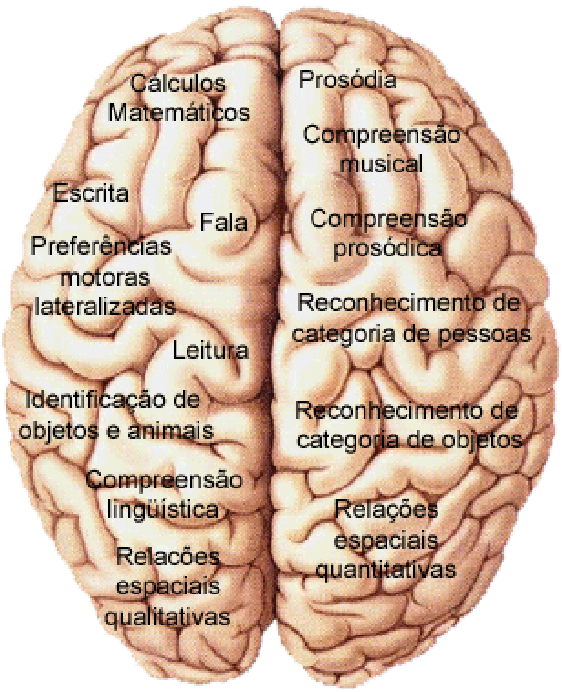


Figura A.1.3 – O cérebro humano e todas suas áreas de atividade.

O termo Inteligência Artificial [Glo.3.2.0.0] tão mal interpretado pelos romances clássicos de ficção, atualmente ganhou interpretação mais abrangente devido aos novos métodos e algoritmos com os quais podemos trabalhar para interpretações, classificações, modelagens e controle.

Nos meios científicos a Inteligência Artificial ou "Artificial Intelligence" (IA ou AI) inicialmente foi a denominação dada à uma estrutura sintática e complexa, um algoritmo de alto nível, que se aproximava da linguagem natural e dotada de recursos que faziam (e ainda fazem) a máquina, aproximar ao comportamento inteligente. Hoje, já são vários os tipos de algoritmos criados e implementados em computador, cada um, eficaz para determinada aplicação, desde funções especializadas como Algoritmos Genéticos (AG) e Sistemas Imunizantes (SI) assim como as generalizações da Linguagem Natural (LN) ou Inteligência Artificial Semântica, a Lógica Fuzzy (LF) e Redes Neurais Artificiais (RNA). Atualmente fazem com que estas máquinas simulem a inteligência animal em suas variadas formas.

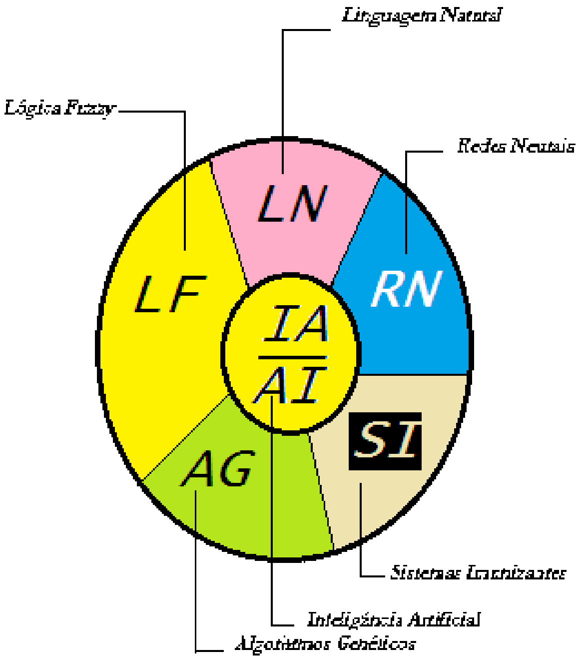


Figura A1.4 – Diagrama dos tipos de Inteligência Artificial.

Algoritmos Genéticos (AG) [Glo.05.0.1] São algoritmos de pesquisa e otimização baseados no princípio da evolução natural e populações genéticas codificadas de indivíduos (Genótipos ou cromossomos) do tipo: $K(t) = K_1(t), ..., K_n(t)$ onde cada indivíduo de $K(t)$ representa uma solução potencial do problema em questão em que são avaliados para se obter suas pertinências quando então, uma nova população é formada em t+1 de indivíduos mais ajustados. Os novos indivíduos (Fenótipos) desta população sofrem transformações devidas a cruzamentos e mutações para formar novas soluções até que no final de um número de gerações, a pesquisa *converge*, sendo os indivíduos finais a melhor solução.

Sistemas Imunizantes (SI) [Glo.05.0.2] apontam em uma direção em que muitos pesquisadores desconhecem que o cérebro, não é o único dotado de iniciativa no corpo animal e uma quantidade considerável de pesquisa tem se concentrado em modelos de inteligência e de aprendizado da forma como elas ocorrem no cérebro humano. Este antropomorfismo ignora sistemas inteligentes que não estão explicitamente relacionados à nossa mente como é o caso do nosso sistema imunológico que se comporta de forma diferente dos neurônios do

cérebro, mas que no final, conseguem atingir eficazmente o seu objetivo. Os sistemas imunológicos se baseiam em agentes e em um princípio de reconhecimento de padrões já existentes e na memorização de novos padrões para neutralizar um agente invasor através de mensagens para células. Quando se neutraliza, temos a solução. Os algoritmos ditos imunizantes seguem este comportamento sendo que para isto é comum o emprego das equações de Riccati [Glo3.2.0.3]. A Equação de Riccati, cujo nome é uma homenagem ao Conde Jacopo Francesco Riccati, é uma equação diferencial ordinária não linear, de primeira ordem, da forma:

$$y' = a_2(x)y^2 + a_1(x)y + a_0(x). \qquad (3.2.1)$$

em que $a_2(x), a_1(x)$ e $a_0(x)$ são funções contínuas num intervalo I e $a_2(x) \neq 0$ em I.

Jacopo Riccati em 1720, propunha duas novas equações diferenciais:

$$y' = \alpha y^2 + \beta x^m \qquad (3.2.2)$$

$$y' = \alpha y^2 + \beta x + \gamma x^2 \qquad (3.2.3)$$

sendo m, α, β e γ constantes e x a variável independente. Foi provavelmente o primeiro documento contendo as formas iniciais da Equação de Riccati. Até então, o principal interesse de Riccati na área de equações diferenciais era nos métodos de solução por separação de variáveis. Possivelmente seu interesse por equações tenha se originado a partir da leitura do livro *De constructione aequationum differentialium primi gradus*, de Gabriele Manfredi, impresso em Bologna em 1707 (Manfredi ocupou a Cátedra de Matemática na Universidade de Bolonha por vários anos). Com relação a equação que leva o seu nome, inicialmente a atenção de Riccati estava voltada para o seguinte problema de natureza geométrica:

Suponha que um ponto de coordenadas $\alpha(x), \beta(y)$ descreva uma trajetória no plano submetida às equações lineares simultâneas de primeira ordem:

$$\alpha' = w_{11}.\alpha + w_{12}.\beta \qquad (3.2.4)$$

$$\beta' = w_{12}.\alpha + w_{22}.\beta \qquad (3.2.5)$$

A questão que Riccati se propôs foi a de determinar o coeficiente angular m da reta tangente a cada ponto da trajetória do ponto:

$$m = \frac{\beta}{\alpha} \qquad (3.2.6)$$

Para solucionar o problema, Riccati teve de resolver preliminarmente a equação de coeficientes constantes $\dot{x} = ax^2 + bx + c$, a qual é normalmente referida como *Equação de Riccati de coeficientes constantes*. Entretanto o próprio Riccati considerou equações com coeficientes tanto constantes quanto variáveis, com especial atenção devotada às equações (3.2.2) e (3.2.3), bem como a $\dot{x} = \alpha t^p x^2 + \beta t^m$ e apresentou diversos métodos para obtenção de soluções para elas. Atualmente, estas são algumas das equações que encontraram

aplicações na área de inteligência artificial e fortuitamente abre um horizonte de pesquisa para sistemas imunizantes.

Lógica Nebulosa ou Lógica Fuzzy (LN ou LF) [Glo3.2.0.4] é a lógica que se utiliza de evasões em uma forma aproximadamente à maneira humana, de lidar com informações imprecisas ou incertas. A teoria de conjuntos nebulosos encontrou uma forma de quantificar a ambiguidade de um elemento x pertencendo a um conjunto A, através de uma função de pertinência $\mu A(x)$, sob uma determinada possibilidade.

Redes Neurais Artificiais (RNA) [Glo3.2.1.5] simulam de forma elementar as redes de neurônios do sistema nervoso e são as ferramentas matemáticas de suporte ao paradigma conexionista na solução de determinados problemas em inteligência artificial.

A utilização de controladores não-convencionais ou baseados em Inteligência Artificial pode ser realizado sem a participação do controle convencional, mas não pode ser vista como uma única solução, sendo incorreta a atitude de trocar sem questionamentos os controladores convencionais (Moderno, Robusto, etc) [Bib.3.2.0.1] por controladores baseados em dados e/ou conhecimento como será feito neste trabalho. Muitas vezes, os controladores convencionais prevalecem (p.ex.PID) em outras, associam-se controladores não–convencionais ou inteligentes aos controles convencionais, aumentando-se a robustez destes.

Mas foi objetivo deste trabalho determinar o quanto é possível termos apenas o controle inteligente baseado na dinâmica do dispositivo de forma que se tornam apropriadas para os propósitos de controle (leis de controle) ou seja, manter a resposta dinâmica de acordo com uma demanda e uma performance do sistema que o mantenha controlável e estável.



"Will Abott drive the Jeep correctly?"

Pretende-se, como foi dito na introdução deste trabalho a abordagem de uma forma de controle não-convencional que igualmente tivesse condições de levar em consideração a dinâmica do dispositivo considerado porque, como foi observado, a tarefa básica em controle de processos consiste em projetar controladores que atuem no sistema de forma que o mesmo responda satisfatoriamente a valores de referência fornecidos ao controlador. Este por sua vez deve atuar na planta, considerando suas características dinâmicas, de forma que a mesma responda com estabilidade às ações dos atuadores.

Sob este ponto de vista, temos diversos tipos de controle inteligentes, entre eles, por uma rede neural, por uma Lógica Nebulosa (ou "Fuzzy Logic") e uma sugestão de maior eficácia, o conjunto das anteriores, ou seja, um controle híbrido Neuro-Fuzzy [Bib.3.2.0.2].

A seguir, apresenta-se uma revisão do desenvolvimento sob cada uma destas formas e na conclusão, a forma de unificá-las.

APD.2. REVISÃO DE REDES NEURAIS FUNDAMENTAÇÃO TEÓRICA

Este ítem descreve a estrutura e comportamento de redes neurais biológicas, de modo a estabelecer uma analogia com os modelos de *redes neurais artificiais*. Na sequência, uma possível definição de rede neural, sua organização, características e histórico da abordagem baseada nestes modelos são apresentados. Também é descrita a forma de representação do conhecimento em uma rede neural e como o conhecimento contido em sua estrutura é utilizado no processo de aprendizado, e consequente realização de previsões.

Redes Neurais tem sido pesquisadas extensivamente nas últimas décadas e aplicadas em muitas áreas. Literatura abundante existe sobre a teoria e aplicações, e nos reportamos a [7] para uma boa introdução.

Para maior clareza, exporemos muito superficialmente, seus pontos fundamentais, enfocando no tipo de rede usada neste trabalho.

As aplicações neurais se enquadram em duas amplas áreas: *aproximação de funções* (e.g., modelamento) e *classificação* (e.g., reconhecimento de padrões). Neste trabalho usamos esta rede como classificadora que para este tipo de reconhecimento vocal é a mais indicada e como veremos, cumpre o seu papel.

APD.2.1. Redes Neurais Biológicas

O cérebro humano é o dispositivo mais complexo estudado pelo homem, sendo ainda muito pobremente compreendido (BEALE1990). Ainda não se tem respostas satisfatórias para as questões mais fundamentais tais como *"o que é mente ?"* e *"como eu penso ?"*. Entretanto, a compreensão geral da operação do cérebro já é possível .

O sistema nervoso humano é formado por uma rede de *neurônios* (figura 3.2.3), unidade básica do cérebro, responsável pelos fenômenos conhecidos como pensamento, emoção e cognição, além da execução das funções sensório-motoras e autônomas (WASSERMAN, 1989). Cada neurônio compartilha várias características com outras células, mas possui capacidades singulares para receber, processar e transmitir sinais eletroquímicos ao longo das fibras nervosas, que compreendem o sistema de comunicação cerebral. (STEVENS, 1985) observa a célula nervosa como uma unidade de processamento analógica independente, capaz de estabelecer a comunicação com o sistema nervoso central.

Conforme (BEALE, 1990), um neurônio é conectado a vários outros neurônios através dos *dendritos* e do *axônio*. Os dendritos - uma complexa rede de prolongamentos - recebem impulsos nervosos de outros neurônios e os conduzem ao *corpo celular* ou *núcleo*. Essas informações são somadas, gerando novos impulsos. Caso o resultado dessa soma exceda um determinado limiar, o axônio transmitirá esse estímulo a outros neurônios através de fenômenos químicos denominados *sinapses*. A força sináptica da conexão neural ao refletir o nível de excitação ou inibição entre neurônios adjacentes, capacita o cérebro humano ao armazenamento do conhecimento e o conseqüente aprendizado.

Através das sinapses os neurônios se unem funcionalmente, formando *redes neurais* (BEALE, 1990).

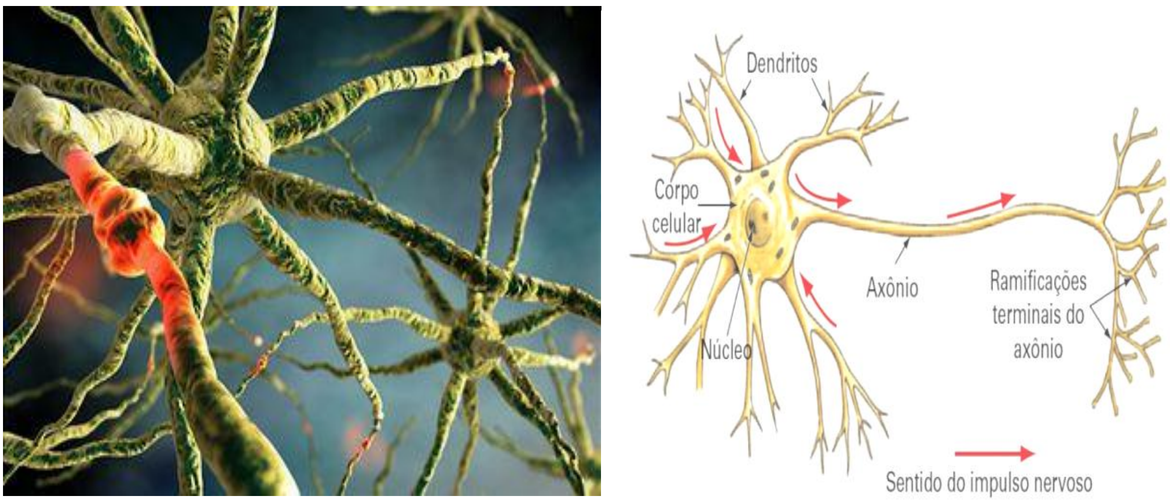


Figura A2.1 Neurônio biológico.
Fonte : (WASSERMAN1989).

APD.2.2. Redes Neurais Artificiais

A abordagem de *redes neurais artificiais* consiste em capturar os princípios básicos de manipulação de informação do cérebro humano e aplicar esse conhecimento na resolução de problemas que exigem aprendizado a partir da experiência (BEALE, 1990).

APD.2.3. Definição

As redes neurais artificiais se fundamentam nos estudos sobre a estrutura do cérebro humano para tentar emular sua forma inteligente de processar informações. Apesar de se desconhecer a maneira pela qual o cérebro manipula informações complexas (BEALE, 1990), sabe-se que a modelagem do conhecimento contido em um problema específico pode ser representada através de inter-conexões entre células nervosas. Estruturalmente, a rede neural artificial, também conhecida por modelo conexionista de computação, se assemelha à rede neural biológica pela composição de seus neurônios e pela conexão entre eles (WASSERMAN, 1989).

"Um modelo conexionista é uma estrutura de processamento de informações distribuída e paralela. Ela é formada por unidades de processamento, comumente chamadas de nós, neurônios ou células, interconectadas por arcos unidirecionais, também chamados de ligações, conexões ou sinapses. Os nós possuem memória local e podem realizar operações de processamento de informação localizada. Cada célula possui uma única saída (axônio), a qual pode se ramificar em muitas ligações colaterais (cada ramificação possuindo o mesmo sinal de saída do neurônio). Todo o processamento que se realiza em cada unidade deve ser completamente local, isto é, deve depender apenas dos valores correntes dos sinais de entrada que chegam dos neurônios através das conexões. Estes valores atuam sobre os valores armazenados na memória local da célula" (HECHT, 1988).

APD.2.4. Histórico das Redes Neurais Artificiais

De acordo com (MÁSSON, 1990), o primeiro modelo formalizado de Redes Neurais Artificiais (RNA) foi proposto por (McCulloch e Pitts, 1943). Essa estrutura abstraía a complexidade da atividade neural em sistemas neurais reais, assim como complicadas características encontradas no corpo de neurônios biológicos, formando a base para a maioria dos modelos conexionistas desenvolvidos posteriormente.

(HEBB, 1949) publica *The Organization of Behavior*, assumindo que a aprendizagem do conhecimento representado em uma rede neural seja alcançada pelo fortalecimento das conexões entre neurônios adjacentes, sempre que esses estiverem excitados.

(Rosenblatt, 1958), Frank criou o *Perceptron*, um modelo cognitivo que consistia de unidades sensoriais conectadas a uma única camada de neurônios de McCulloch e Pitts, capaz de aprender tudo o que pudesse representar. Rosenblatt demonstrou que se fossem acrescentadas, sinapses ajustáveis, as redes neurais de McCulloch e Pitts, poderiam ser treinadas para classificar padrões em classes linearmente separáveis, convergindo em um número limitado de passos (MÁSSON1990).

No início da década de 60, (Widrow e Hoff, 1960) publicam um artigo no qual especificam um neurônio artificial baseado no modelo de McCulloch e Pitts, denominado *Adaline*. Conforme (MÁSSON, 1990), a importância desse modelo está associada à regra de aprendizagem proposta, a regra Delta.

A publicação de *Perceptrons* de (Minsky e Papert, 1969), expôs as limitações do modelo de Rosenblatt, provando que tais redes não são capazes de resolver uma ampla classe

de problemas entre eles o Ou-Exclusivo devido às restrições de representação (MÁSSON, 1990) (WASSERMAN, 1989). O impacto desta publicação foi devastador, praticamente desaparecendo o interesse em redes neurais artificiais nos anos seguintes. Somente poucos pesquisadores como Malsburg, Grossberg, Kohonen e Anderson permaneceram concentrando suas atividades na abordagem conexionista.

A partir dos anos 80, com o avanço da tecnologia e o fracasso da escola simbolista na solução de determinados tipos de problemas, as redes neurais artificiais passaram a atrair substancial atenção novamente (WASSERMAN, 1989).

Em 1982, a introdução do modelo conexionista conhecido pelo nome de seu idealizador, John Hopfield, permitiu esclarecer, pelas suas características computacionais e estabilidade, boa parte das dúvidas até então existentes em relação ao processo dinâmico executado por certas redes neurais. No mesmo ano, Kohonen publicou um artigo descrevendo uma rede neural artificial baseada em auto-organização e nas características de aprendizado adaptativo do cérebro humano (MÁSSON, 1990).

(Hinton e Seynowsky, 1983), estenderam o modelo de Hopfield com a incorporação de dinâmica estocástica. Este modelo de rede neural passou a ser conhecido como Máquina de Boltzmann (WASSERMAN, 1989).

Cerca de dois anos mais tarde, Rumelhart, Hinton e Williams aperfeiçoaram a ideia do Perceptron, criando o algoritmo *Backpropagation* (RUMELHART, 1986), levando a uma explosão de interesse em redes neurais. Este algoritmo foi aplicado em uma grande variedade de problemas, como na identificação da estrutura de proteínas, hifenização de palavras em inglês, ***reconhecimento da fala***, compressão de imagens e previsão de séries temporais (MÁSSON, 1990). O sucesso deste algoritmo estimulou o desenvolvimento de muitas pesquisas em redes neurais artificiais e de uma variedade de modelos cognitivos.

APD.2.5. Topologia das Redes Neurais Artificiais

De acordo com (MÁSSON, 1990), a topologia de uma rede neural artificial pode ser expressa através de um grafo dirigido com pesos G = (V, A, W), onde V corresponde a um conjunto de vértices, A a um conjunto de arcos dirigidos e W a um conjunto de pesos para esses arcos. Cada vértice no grafo representa uma unidade de processamento.

Estas consistem de uma rede de elementos de processamento, simples e não-lineares, chamados Perceptrons (*neurônios artificiais*), altamente interconectados entre si. Alguns destes são acessíveis para estímulos de entrada, outros permitindo o acesso às suas saídas, e os demais inacessíveis formando várias camadas escondidas (figura 3.2.4).

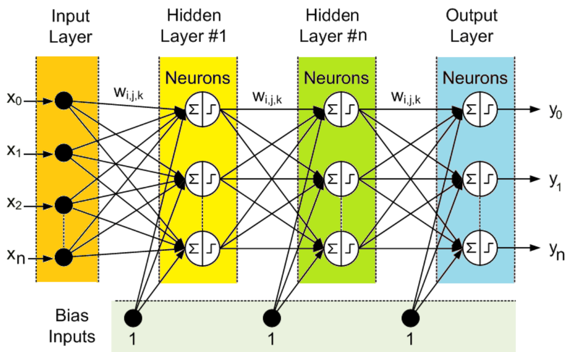


Figura A2.2 – Rede MultiLayer Perceptron (MLP)

O paralelismo, a não - linearidade e o alto grau de conectividade, capacitam às redes neurais a implementação de qualquer mapeamento de função, se esta existe. Suas

características mais importantes são: (1) o aprendizado de um mapeamento através de exemplos, sem a introdução explícita de conhecimentos, exceto a escolha apropriada dos dados e (2) ser capaz de *generalizar* durante a operação (após treinamento), ou seja, responder corretamente a estímulos desconhecidos, desde que pertençam ao mesmo domínio de estímulos para a qual foi treinada.

O paradigma mais utilizado, por sua aplicabilidade universal, é o "Perceptron multicamadas" (MLP), ou rede neural *"Feedforward"*, composta por *camadas* de neurônios das entradas às saídas, onde interconexões são permitidas somente das saídas dos neurônios de uma camada para as entradas dos da seguinte. O neurônio mais comum tem o seguinte modelo matemático:

$$y = f\left(\sum_{k=1}^{n} w_k \times x_k - b\right) = f\left(\sum_{k=0}^{n} w_k \times x_k\right) \qquad (3.2.7)$$

Onde:

y é a saída, x_k (k=1,2,..,n) são os elementos do vetor n-dimensional de entradas, b é um termo de *polarização* (transformado para o peso w_0 correspondente à uma entrada fictícia x_0 = -1 no lado mais à direita de (1)), w_k são *pesos* e f é uma função chamada *função de ativação*.

A rede armazena o conhecimento desejado para a aplicação de forma distribuída, através dos valores dos pesos. Isto é realizado por um processo de treinamento chamado *aprendizado supervisionado*, para o qual um número estatisticamente significante de pares entrada(s)-saída(s) de exemplos (o *conjunto de aprendizagem ou treinamento*) é apresentado à rede de forma iterativa. O algoritmo de aprendizado mais popular associado à rede neural "Feedforward" é o de *retropropagação do erro* com a técnica de minimização chamada *gradiente descendente*.

As funções de ativação mais comumente empregadas pertencem à classe das *sigmoides*, porque são monotônicas, contínuas, apresentam a não-linearidade necessária para mapeamento universal, e tem o efeito de controle automático de ganho.

A rede é treinada várias vezes para diferentes combinações de parâmetros tais como, número de neurônios para cada camada escondida, número de camadas escondidas e tipo de função de ativação (sua *topologia*), geralmente como tentativa-e-erro. Uma rede demasiado pequena para a aplicação não será capaz de aprender satisfatoriamente (o algoritmo de aprendizado não converge para a meta estipulada), enquanto uma rede demasiado grande simplesmente decorará os exemplos do conjunto de aprendizagem, perdendo generalização. Uma maneira eficaz de garantir generalização é preparar outro conjunto de exemplos, chamado *conjunto de validação*, aproximadamente do mesmo tamanho do conjunto de aprendizagem, sobre o qual são realizadas simulações de operação durante o treinamento com o conjunto de aprendizado. Quando o erro sobre o conjunto de validação começar a aumentar, significa que a rede está no limiar de se especializar nos exemplares do conjunto de treinamento, a partir do qual o treinamento deteriorará a generalização. Uma vez obtido um treinamento satisfatório, os pesos são fixados, e a rede estará pronta para operação onde responderá a estímulos desconhecidos. É boa prática também utilizar um terceiro conjunto de dados, denominado *conjunto de teste*, para verificação definitiva do desempenho da rede treinada.

APD.2.6. Camadas

As pesquisas em redes neurais artificiais levaram ao desenvolvimento dos mais diversos modelos cognitivos, cada qual com suas particularidades e adequados a um tipo de situação.

A estruturação de uma rede neural em camadas é uma importante característica topológica desses modelos (MÁSSON, 1990). Em uma rede neural estruturada em camadas, o conjunto de vértices V pode ser particionado em vários subconjuntos disjuntos $V = V^{(0)} V^{(1)} \ldots V^{(L)}$ de modo que as unidades de processamento da camada *l* somente apresentem conexões com as unidades das camadas *l+1* e *l-1*, onde *L* corresponde ao número de camadas da rede neural artificial. (YOUNGOHC, 1991a) define uma rede plenamente conectada ou " *fully connected*" como sendo onde cada unidade de processamento da camada *l* estabelece conexão com todas as unidades de processamento da camada *l+1*.

Quanto ao número de camadas, as redes neurais podem ser dispostas em uma única camada, configuração mais simples de uma rede neural, ou em múltiplas camadas (BEALE, 1990) (WASSERMAN, 1989).

(LIPPMAN, 1987) ainda classifica as redes neurais artificiais em redes cíclicas, também chamadas de redes recorrentes, e redes acíclicas ou não recorrentes. A arquitetura de uma rede neural cíclica difere da acíclica por apresentar conexões entre as unidades de processamento pertencentes à mesma camada ou entre unidades de processamento de camadas diferentes cujas saídas passam a ser entradas na camada anterior. As redes recorrentes podem exibir propriedades muito similares à memória de curto termo dos seres humanos, onde o estado da saída da rede depende em parte da entrada anterior.

APD.2.7. Conexões e Pesos

Os arcos do grafo são chamados de conexões e representam as sinapses entre os neurônios artificiais. A cada conexão no grafo está associado um peso $w_{ij}^{(l)}$, em analogia às sinapses de um modelo conexionista biológico, representando a força de ligação entre as unidades de processamento $v_i^{(l)}$ e $v_j^{(l-1)}$, onde *i* e *j* correspondem à posição - respectivamente nas camadas *l* e *l-1* -, que essas unidades ocupam na rede (MÁSSON, 1990).

Conexões com pesos positivos, chamadas excitadoras, indicam o reforço na ativação do neurônio $v_i^{(l)}$; sinapses com pesos negativos, chamadas inibidoras, indicam a inibição na ativação do neurônio $v_i^{(l)}$. Assim, os neurônios artificiais, distribuídos no espaço e ligados por conexões, trocam sinais excitatórios ou inibitórios, competindo ou cooperando entre si. O comportamento inteligente emerge então, da ação simultânea dessa coletividade, sem a necessidade de elementos centralizadores (MÁSSON, 1990).

APD.2.8. Unidades Visíveis e Unidades Ocultas

A interface da rede neural artificial é definida pelas unidades de entrada (V_I), unidades de saída (V_O) e pelas unidades ocultas (V_H) (MÁSSON, 1990) (YOUNGOHC, 1991a).

$$V = V_I \, V_O \, V_H$$

O conjunto de unidades de entrada e de saída, representam as unidades visíveis da rede, sendo dependentes da aplicação que se quer modelar (FREEMAN, 1992). Em redes neurais estruturadas em camadas, a camada de entrada é tipicamente $V^{(0)}$ e a camada de saída, $V^{(L)}$.

As unidades ocultas são utilizadas para modificar os dados de entrada, de modo a suportar qualquer função requerida para a entrada ou saída (YOUNGOHC, 1991b), impondo uma representação intermediária adicional dos dados de entrada para a saída desejada. Através das unidades ocultas, o modelo conexionista consegue representar abstrações que não poderiam ser diretamente realizadas a partir das unidades de entrada (JONES, 1987).

APD.2.9. Representação do Conhecimento em uma Rede Neural Artificial

O domínio do conhecimento de um problema é representado em um modelo conexionista através das unidades de processamento, que abstraem a estrutura e o comportamento dos neurônios biológicos (BEALE, 1990).

A figura 3.2.5 ilustra a estrutura de uma unidade de processamento de um neurônio artificial.

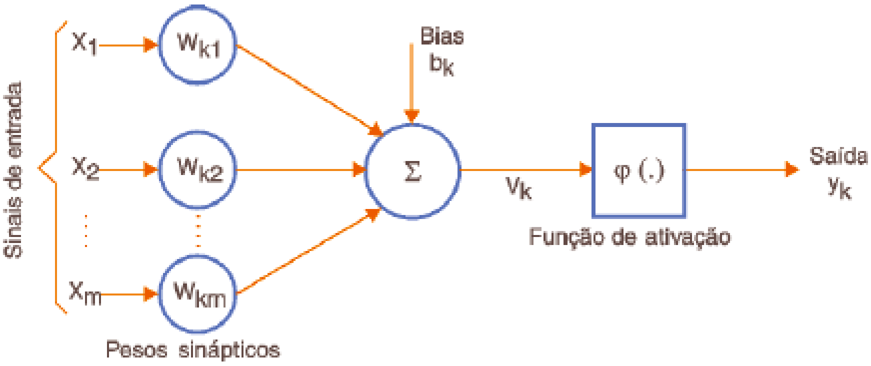


Figura A2.3 Unidade de processamento.

Uma unidade de processamento $v_i^{(l)}$ possui entradas $x_1^{(l-1)}$, $x_2^{(l-1)}$,..., $x_n^{(l-1)}$, que correspondem aos estados dos neurônios $v_j^{(l-1)}$ com os quais está conectada.

A partir dessas entradas e do conjunto de pesos sinápticos $w_{ij}^{(l)}$, que refletem a força da unidade $v_j^{(l-1)}$ sobre a unidade $v_i^{(l)}$, é calculado o potencial *net* do neurônio $v_i^{(l)}$. Esse potencial no tempo *t* é determinado por uma *regra de propagação*, que geralmente equivale à soma linear da multiplicação das entradas pelos pesos (MÁSSON, 1990), conforme equação (3.2.8).

$$net_i^{(l)}{}_{(t)} = \sum_{j=1}^{n^{(l-1)}} w_{ij}^{(l)} x_j^{(l-1)}{}_{(t)} - \theta_i^{(l-1)}{}_{(t)} \tag{3.2.8}$$

Onde:

$x_j^{(l-1)}$ é o estado da j-ésima unidade,

$w_{ij}^{(l)}$ é a força sináptica entre a i-ésima unidade e a j-ésima unidade e,

${}_i^{(l)}$ é o limiar da i-ésima unidade, representando a força que as entradas das unidades conectadas à unidade $v_i^{(l)}$ precisam atingir para ativar esta unidade.

O potencial é modificado pela aplicação de uma *função de ativação g*, determinando o estado da unidade $v_i^{(l)}$ no instante *t+1*.

$$x_i^{(l)}{}_{(t+1)} = g\left(net_i^{(l)}{}_{(t)}\right) \tag{3.2.9}$$

De acordo com (MÁSSON, 1990], a função de ativação corresponde a um limiar que restringe a propagação do impulso "nervoso" à transposição de um certo nível de atividade, mapeando o potencial da unidade de processamento $v_i^{(l)}$ para um intervalo pré-especificado de saída.

Temos várias possíveis funções de ativação podendo-se citar :

A função *degrau* (figura 3.2.6a) ilustra que admite valor +1 se o potencial da unidade de processamento $v_i^{(l)}$ for positivo e -1, caso contrário.

A função *linear* (figura 3.2.6b) é limitada ao intervalo [-y,+y] definida por:

$$g(net_i^{(l)}{}_{(t)}) = \begin{cases} +y, \text{ se } net_i^{(l)}{}_{(t)} \geq y \\ net_i^{(l)}{}_{(t)}, \text{ se } \left|net_i^{(l)}{}_{(t)}\right| < y \\ -y, \text{ se } net_i^{(l)}{}_{(t)} \leq y \end{cases}$$

Onde: y representa o ponto de saturação da função.

A função *logística (sigmoide)* (figura 3.2.6c), também conhecida por função *logística*, é expressa matematicamente como:

$$g(net_i^{(l)}{}_{(t)}) = \frac{1}{1+e^{-net_i^{(l)}{}_{(t)}}} \qquad (3.2.10)$$

sendo uma função contínua, monotonicamente crescente e que gera valores graduais e não lineares no intervalo [0,1].

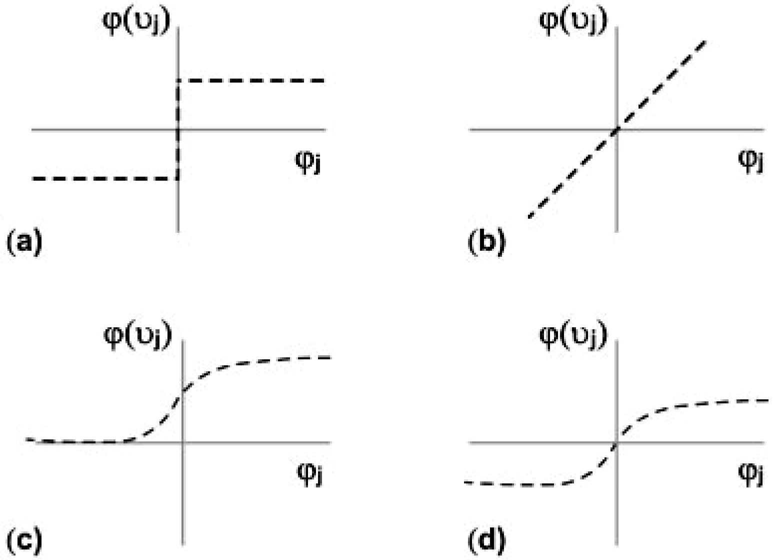


Figura A2.4 – Representaçãográfica de diferentes funções de ativação sendo que em (a) função degrau; em (b) função linear; (c) função logística e; (d) função tangente hiperbólica.

Outra função de ativação comumente utilizada é a *tangente hiperbólica* (figura 3.2.6d) (WASSERMAN, 1989). Sua curva é similar à função sigmoide, mas simétrica na origem. Essa função é frequentemente empregada por biólogos como um modelo matemático de ativação da célula nervosa.

Mesmo conhecendo completamente o comportamento de cada neurônio individual, a composição de várias unidades de processamento em uma estrutura de rede manifesta reações imprevisíveis. Desta forma, é a reunião do estado de ativação de todas as unidades de processamento que especifica o que está sendo representado na rede neural artificial em um determinado instante. É essa emergência de propriedades de conjunto que determina o interesse e a complexidade dos modelos conexionistas.

APD.2.10. Características das Redes Neurais Artificiais

Por serem baseadas nas Redes Neurais Biológicas, as Redes Neurais Artificiais apresentam um surpreendente número de características observadas no processo cognitivo

humano (WASSERMAN, 1989), como o aprendizado pela experiência, a generalização a partir de exemplos e a abstração de características essenciais de informações que contém fatos irrelevantes.

APD.2.11. Aprendizado

(CARBONELL, 1989) define o conceito de aprendizado como a habilidade de realizar tarefas novas que não podiam ser realizadas anteriormente, ou melhorar a realização de tarefas antigas, como resultado de mudanças produzidas pelo processo de aprendizado.

As RNAs podem modificar seu comportamento em resposta aos estímulos produzidos pelo ambiente, regulando a força da conexão entre unidades de processamento adjacentes pela adaptação dos *pesos sinápticos*, reconhecendo as informações apresentadas às suas unidades visíveis (WASSERMAN, 1989).

APD.2.12. Generalização

Segundo (WASSERMAN, 1989), um modelo conexionista é sensível às variações que podem ocorrer em informações procedentes de suas unidades de entrada, reconhecendo ruído e distorção. A capacidade da rede em se adaptar às novas situações, gerando valores de saída consistentes com os esperados, é vital para a aplicabilidade do modelo em um ambiente do mundo real.

Embora a maioria das pesquisas em redes neurais artificiais tenham concentrado seus esforços na redução dos tempos de aprendizagem, a característica mais importante de um modelo conexionista é a habilidade em generalizar sobre o domínio do problema (REFENES, 1993b).

O bom desempenho da generalização depende, entre outros fatores, do número de parâmetros livres da RNA, Rede Neural Artificial (LECUN, 1989). É desejável diminuir o tamanho das conexões sem, entretanto, reduzir o tamanho da rede ao ponto onde não se possa computar a função desejada.

APD.2.13. Abstração

Alguns modelos de redes neurais artificiais são capazes de abstrair a essência do conjunto de dados a elas apresentados (WASSERMAN, 1989), permitindo, dessa forma, a classificação ou reconhecimento de padrões incompletos.

APD.2.14. Aprendizado da Rede Neural Artificial

Dentre todas as características das redes neurais artificiais, nenhuma desperta tanto interesse quanto a sua habilidade em realizar o aprendizado (WASSERMAN, 1989).

Conforme (MÁSSON, 1990), o aprendizado em um modelo de RNAs é decorrente do treinamento da rede através da apresentação de padrões às suas unidades visíveis.

O objetivo do treinamento consiste em atribuir os *pesos sinápticos* com valores apropriados, de modo a produzir o conjunto de saídas desejadas ou ao menos consistentes com um intervalo de erro estabelecido (FREEMAN, 1992). Desta forma, o processo de aprendizado subsiste na busca de um espaço de pesos pela aplicação de alguma regra que defina esta aprendizagem (MÁSSON, 1990).

Em geral, as regras de aprendizado podem ser consideradas variantes da Regra de Hebb (HEBB, 1949). Na essência Hebb propõe que, a sinapse conectando dois neurônios seja reforçada sempre que ambos neurônios estiverem ativos.

Uma rede neural artificial que tenha a regra de Hebb como regra de aprendizado modifica os pesos sinápticos entre as conexões das unidades de processamento $v_i^{(l)}$ e $v_j^{(l-1)}$ proporcionalmente ao produto dos níveis de excitação desses neurônios, conforme equação (3.2.11).

$$\Delta w_{ij}^{(l)} = \eta . x_i^{(l)} . x_j^{(l-1)} \tag{3.2.11}$$

Onde:

$\Delta w_{ij}^{(l)}$ corresponde a alteração no valor do peso $w_{ij}^{(l)}$ e é uma constante de proporcionalidade que reflete a evolução do processo de aprendizado pela busca no espaço de pesos.

Como adaptação à regra de Hebb, a regra Delta modifica os pesos de acordo com a variação entre a saída desejada e a observada no treinamento (MÁSSON, 1990).

A equação (3.2.12) atualiza os pesos associados aos arcos da rede neural artificial pela aplicação da regra Delta abaixo:

$$\Delta w_{ij}^{(l)} = \eta \ \delta \ x_i^{(l)} \tag{3.2.12}$$

com $\delta = (d_i^{(l)} - x_i^{(l)})$, onde $d_i^{(l)}$ representa a saída desejada para a unidade de processamento $v_i^{(l)}$.

APD.2.15. Estratégias de Aprendizado

Uma rede neural artificial deve ser ajustada para que a aplicação de um conjunto de entradas produza a saída desejada. Esse ajustamento, obtido pelo treinamento da rede, pode ser feito das seguintes formas (MÁSSON, 1990) (WASSERMAN, 1989) :

Sem treinamento : Os valores dos pesos sinápticos são estabelecidos explicitamente.

Treinamento supervisionado : A rede é treinada pela apresentação dos vetores de entrada e seus respectivos vetores de saída, chamados de pares de treinamento.

Treinamento não-supervisionado : O treinamento consiste da apresentação apenas dos vetores de entrada, a partir dos quais são extraídas as características desse conjunto de padrões, agrupando-os em classes. O treinamento não supervisionado pode ser observado como um processo autônomo ou auto-organizável.

APD.2.16. Algoritmo de Aprendizado *Backpropagation*

Por um intervalo de muitos anos (1960-1980), não se teve um algoritmo eficiente para treinar redes neurais artificiais de múltiplas camadas. Desde que as redes de uma única camada se mostraram limitadas naquilo que poderiam representar e, portanto, no que poderiam aprender, o desenvolvimento de modelos cognitivos deixou de ser um campo atraente e poucas pesquisas foram realizadas na área.

O algoritmo "*Backpropagation*" (retropropagação), proposto por Werbos, Parker e Rummelhart, fez ressurgir o interesse em Redes Neurais Artificiais, sendo o algoritmo de aprendizado mais largamente utilizado (MÁSSON, 1990) (REFENES, 1993c).

Conforme (BEALE, 1990), o "*Backpropagation*" pode ser visto como uma generalização do método Delta para redes neurais de múltiplas camadas. Ao se apresentar um determinado padrão de entrada a uma rede neural não treinada e o respectivo padrão de saída, uma saída aleatória é produzida. A partir da saída produzida pela rede é calculado um erro, representando a diferença entre o valor obtido e o desejado. O objetivo consiste, então, em

reduzir continuamente o erro até um determinado valor aceitável. Isto é alcançado pelo ajuste dos pesos entre as conexões dos neurônios pela aplicação da regra Delta Generalizada, que calcula o erro para alguma unidade particular e propaga esse erro para a camada anterior. Cada unidade tem seus pesos ajustados de modo a minimizar o erro da rede.

A minimização do erro no algoritmo "*Backpropagation*" é obtida pela execução do *gradiente decrescente* na superfície de erros do espaço de pesos, onde a altura para qualquer ponto no espaço de pesos corresponde à medida do erro. O ajuste dos pesos inicia nas unidades de saída, onde a medida do erro está disponível, e procede com a retropropagação desse erro entre as camadas, ajustando os pesos até que a camada das unidades de entrada tenha sido processada. Para as unidades de saída, como são conhecidos os valores desejados e obtidos, o ajuste dos pesos sinápticos é relativamente simples; para as unidades das camadas ocultas, o processo não é tão trivial. Intuitivamente, as unidades ocultas que apresentarem erros grandes devem ter suas conexões bastante alteradas, enquanto que a mudança nos pesos daquelas que tiverem suas saídas muito próximas das desejadas deverá ser pequena. Na realidade, os pesos para um neurônio particular devem ser ajustados na proporção direta ao erro da unidade de processamento a qual está conectado. Essa é a razão pela qual a retropropagação dos erros através da rede permite o correto ajuste dos pesos sinápticos entre todas as camadas do modelo conexionista.

Assim, é possível identificar duas fases distintas no processo de aprendizagem do "*Backpropagation*": aquela onde as entradas se propagam entre as camadas da rede, da camada de entrada até a camada de saída, e aquela em que os erros são propagados na direção contrária ao fluxo de entrada.

APD.2.17. Etapas do Algoritmo "*Backpropagation*"

Seja o conjunto de treinamento D = (d_I^λ ,) : = 1, ..., , onde d_I^λ são os vetores de entrada, d_O^λ os correspondentes vetores de saída e o número de vetores do conjunto de treinamento.

Conforme (FREEMAN, 1992) e (MÁSSON, 1990), o objetivo do treinamento consiste em ensinar à rede o mapeamento de todo vetor de entrada para o respectivo vetor de saída, isto é, estabelecer x_o (w , d_I^λ) = d_O^λ, 1 , pela configuração de valores apropriados para os pesos das conexões da rede.

O ajustamento desses pesos, executado sempre que a saída desejada d_o e a saída real x_o não forem coincidentes, é feito pela aplicação do método *gradiente decrescente* (equação (3.2.13)).

$$w_{ij}^{(l)}{}_{(t+1)} = w_{ij}^{(l)}{}_{(t)} + \Delta w_{ij}^{(l)} \qquad (3.2.13)$$

com $\Delta w_{ij}^{(l)} = -\eta \dfrac{\partial \varepsilon}{\partial w_{ij}^{(l)}}$, onde η é a taxa de aprendizagem e $\Delta w_{ij}^{(l)}$, a medida do erro.

O erro é definido como uma soma sobre todos os padrões de treinamento, conforme equação (3.2.14).

$$\varepsilon = \sum_{\lambda=1}^{\Lambda} \varepsilon^{\lambda} \qquad (3.2.14)$$

Como medida de erro para um padrão de treinamento específico tem-se na equação (3.2.15):

$$\varepsilon^{\lambda} = \frac{1}{2}\sum_{j} (d_{o,j}^{\lambda} - x_{o,j}^{\lambda})^2 \qquad (3.2.15)$$

Onde:

$d_{o,j}^{\lambda}$ é o j-ésimo componente do vetor de saída d_o^{λ}. O aprendizado é reduzido, portanto, à minimização da medida de erro na equação (3.2.15).

As etapas do "*Backpropagation*" podem ser sumarizadas nos seguintes passos :

Inicializar o conjunto de pesos W com valores pequenos e aleatórios;

Apresentar o vetor de entrada do padrão de treinamento $x^{(0)} = d_I^{\lambda}$;

Calcular a resposta da rede $x^{(L)} = x^{(L)}(w, x^{(0)})$;

Calcular o erro para as unidades de saída

$$\delta_i^{(L),\lambda} = g'(net_i^{(L)})\ (d_{o,i}^{\lambda} - x_i^{(L)}) \qquad (3.2.16)$$

Onde:

g' é a derivada da função de ativação em relação à $net_i^{(L)}$;

Calcular o erro para as unidades ocultas

$$\delta_i^{(l),\lambda} = g'(net_i^{(l)}) \sum_{k} \delta_k^{(l+1),\lambda}\ w_{ki}^{(l+1)} \qquad (3.2.17)$$

Atualizar os pesos de todas as camadas

$$\Delta w_{ij}^{(l),\lambda} = \eta\ \delta_i^{(l),\lambda}\ x_j^{(l-1)} \qquad (3.2.18)$$

Os padrões de treinamento são apresentados sucessivamente às unidades visíveis da rede neural artificial, até que um erro aceitável seja alcançado ou enquanto um número determinado de iterações não for satisfeito. O último conjunto de pesos observado entre as conexões das células é então mantido para testar a habilidade da rede em mapear a função de entrada para saída e a conseqüente validação do modelo de redes neurais artificiais (FREEMAN, 1992).

APD.2.18. Considerações sobre o Algoritmo "*Backpropagation*"

O desempenho do algoritmo de aprendizagem "*Backpropagation*" está condicionado à modelagem adotada na RNA e ao conjunto de dados utilizados no processo de ajuste dos pesos sinápticos entre as conexões da rede.

APD.2.19. Dados para Treinamento

Conforme James Freeman (FREEMAN, 1992), não existe critério específico para seleção dos vetores de treinamento. É possível utilizar todos os dados disponíveis no treinamento do modelo conexionista, embora apenas um subconjunto desses dados talvez seja suficiente para que esse processo seja executado com sucesso. Os dados restantes podem ser usados para avaliar a capacidade de generalização do "*Backpropagation*" no mapeamento de entradas nunca encontradas no treinamento para saídas consistentes.

APD.2.20. Tamanho da Rede Neural Artificial

(REFENES, 1993c) coloca que o número de unidades de processamento das camadas de entrada e saída é usualmente determinado pela aplicação. No caso das camadas ocultas, a relação não é tão transparente. O ideal é utilizar o menor número possível de unidades ocultas para que a generalização não fique prejudicada (RUMELHART, 1986). Se o número de neurônios ocultos for muito grande, a rede acaba memorizando os padrões apresentados durante o treinamento. Contudo, se a arquitetura das camadas ocultas possuir unidades de processamento em número inferior ao necessário, o algoritmo "*Backpropagation*" pode não conseguir ajustar os pesos sinápticos adequadamente, impedindo a convergência para uma solução.

A experiência ainda é a melhor indicação para a definição da topologia de um modelo conexionista (SURKAN, 1990).

APD.2.21. Pesos e Parâmetros de Aprendizado

(FREEMAN, 1992) sugere que os pesos das conexões entre as camadas de uma rede neural sejam inicializados com valores aleatórios e pequenos para que se evite a saturação da função de ativação e a consequente incapacidade de realizar a aprendizagem.

À medida que o treinamento evolui, os pesos sinápticos podem passar a assumir valores maiores, forçando a operação dos neurônios na região onde a derivada da função de ativação é muito pequena. Como o erro retropropagado é proporcional a esta derivada, o processo de treinamento tende a se estabilizar, levando a uma paralisação da rede sem que a solução tenha sido encontrada. Isto pode ser evitado pela aplicação de uma taxa de aprendizagem menor. Teoricamente, o algoritmo de aprendizado exige que a mudança nos pesos seja infinitesimal (RUMELHART, 1986). Entretanto, a alteração dos pesos nessa proporção é impraticável, pois implicaria em tempo de treinamento infinito. Em vista disso, é recomendável que a taxa de aprendizado assuma valor maior no início do treinamento e, à medida que se observe decréscimo no erro da rede, essa taxa também seja diminuída. Diminuindo progressivamente a taxa de atualização dos pesos, o *Gradiente Descendente* está apto a alcançar uma solução melhor (BEALE, 1990).

Outra maneira de aumentar a velocidade de convergência da rede neural artificial treinada pelo algoritmo "*Backpropagation*" é a adoção de um método chamado "*momentum*" (BEALE, 1990) (FREEMAN, 1992) (RUMELHART, 1986) (WASSERMAN, 1989). O propósito desse método consiste em adicionar, quando do cálculo do valor da mudança do peso sináptico, uma fração proporcional à alteração anterior. Assim, a introdução desse termo na equação de adaptação dos pesos tende a aumentar a estabilidade do processo de aprendizado, favorecendo mudanças na mesma direção. A equação (3.3.12) especifica o ajuste das conexões entre unidades de processamento pela aplicação do termo "*momentum*".

$$\Delta w_{ij}^{(l),\lambda}{}_{(t+1)} = \eta\, \delta_i^{(l),\lambda}\, x_j^{(l-1)} + \alpha \left(w_{ij}^{(l),\lambda}{}_{(t)} - w_{ij}^{(l),\lambda}{}_{(t-1)}\right) \qquad (3.2.19)$$

Onde:

α representa o termo "*momentum*", $0 < \alpha < 1$.

APD.2.22. Mínimo Local

O "*Backpropagation*" utiliza o método do *Gradiente Descendente* para ajustar os pesos entre as sinapses, seguindo a curva da superfície dos erros em direção a um ponto mínimo (WASSERMAN, 1989). Superfícies de erros convexas, por apresentarem um único mínimo, permitem que este método atinja o mínimo global. Nas superfícies de erros não convexas e altamente convolutas, normalmente encontradas em problemas práticos, a solução alcançada

pode não ser a ótima. Nestes casos, deverá ser utilizado algum algoritmo de otimização global.

Assim que um mínimo é encontrado, seja global ou local, o aprendizado cessa (FREEMAN, 1992). Se a rede alcançar um mínimo local (figura 3.2.7), do seu ponto de vista limitado, todas as direções em sua volta representam valores maiores que o alcançado e, conseqüentemente, a convergência para o mínimo global não é atingida. Nesse caso, a magnitude do erro da rede pode ser muito alta e, portanto, inaceitável.

Caso a rede neural encerre o aprendizado antes que uma solução satisfatória seja obtida, o redimensionamento do número de unidades ocultas ou da taxa de aprendizagem e do termo *momentum* podem ser suficientes para resolver o problema. De acordo com [FREEMAN1992], outra possibilidade para se tentar encontrar o mínimo global é realizar o treinamento a partir de um conjunto de pesos inicial diferente daquele utilizado anteriormente.

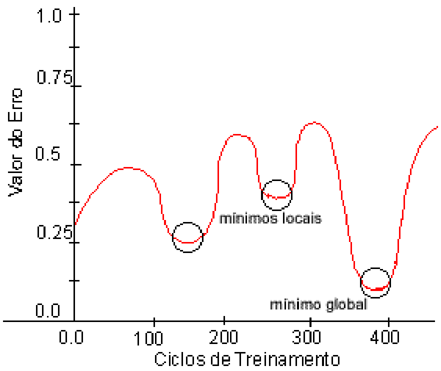


Figura A2.5 - Mínimo local.
Fonte : [KOVÁCS1996].

Através da figura A2.5, que ilustra um corte em uma superfície de erros 3D hipotética no espaço de pesos, é possível observar pontos de mínimos locais à esquerda e à direita, um mínimo global, os valores destes são maiores que esse mínimo global.

APD.3. REVISÃO DE LÓGICA FUZZY NEBULOSA FUNDAMENTAÇÃO TEÓRICA

Aristóteles, filósofo grego (384 - 322 a.C.), foi o fundador da ciência da lógica, e estabeleceu um conjunto de regras rígidas para que conclusões pudessem ser aceitas logicamente válidas. O emprego da lógica de Aristóteles levava a uma linha de raciocínio lógico baseado em premissas e conclusões. Como um exemplo: se é observado que "todo ser vivo é mortal" (premissa 1), a seguir é constatado que "Sarah é um ser vivo" (premissa 2), como conclusão, temos que "Sarah é mortal".

Desde então, a lógica Ocidental, assim chamada, tem sido binária, isto é, uma declaração é falsa ou verdadeira, não podendo ser ao mesmo tempo parcialmente verdadeira e parcialmente falsa. Esta suposição e a lei da não contradição cobrem todas as possibilidades, formam a base do Pensamento Lógico Ocidental.

A *Lógica Fuzzy* (ou *Lógica Nebulosa*) viola estas suposições. Um sim ou um não como resposta a estas questões é, na maioria das vezes, incompleta. Na verdade, entre a certeza de ser e a certeza de não ser, existem infinitos graus de incerteza. Esta imperfeição intrínseca à informação representada numa linguagem natural tem sido tratada matematicamente no passado com o uso da teoria das probabilidades. Contudo, a *Lógica Fuzzy*, com base na teoria dos *Conjuntos Fuzzy* (ou *Conjuntos Nebulosos*) que é uma teoria de possibilidades, tem se

mostrado mais adequada para tratar imperfeições da informação do que a teoria das probabilidades.

A *Lógica Fuzzy* encontra-se entre as técnicas mais atuais de *Inteligência Artificial*, também conhecida como já foi dito por *Conjuntos Fuzzy*. Este termo, a princípio, nos convida a pensar em algo confuso (nebuloso), porém, atualmente, é bastante direto. Essa técnica, muito usada no Japão, e mais recentemente em todo oriente dos países em desenvolvimento é fruto da tão estudada quinta geração dos computadores, uma geração que morreu antes mesmo de nascer.

A *Lógica Fuzzy* consiste em aproximar a decisão computacional da decisão humana, tornando as máquinas mais capacitadas a seu trabalho. Isto é feito de forma que a decisão de uma máquina não se resuma apenas a um "sim" ou um "não", mas também tenha decisões "abstratas", do tipo "um pouco mais", "talvez sim", e outras tantas variáveis que representem as decisões humanas. É um modo de interligar inerentemente processos analógicos que se deslocam através de uma faixa contínua para um computador digital que podem ver coisas com valores numéricos bem definidos (valores discretos).

Uma das principais potencialidades da *Lógica Fuzzy*, quando comparada com outros esquemas que tratam com dados imprecisos como redes neurais, é que suas bases de conhecimento, as quais estão no formato de regras de condições, são fáceis de examinar e entender. Este formato de regra também torna fácil a manutenção e a atualização da base de conhecimento.

O conceito de *Conjunto Fuzzy* foi introduzido em 1965 (Zadeh, 1965a), por Lotfali Askar Zadeh (Universidade da Califórnia, Berkeley). A ele é atribuído o reconhecimento como grande colaborador do Controle Moderno. Em meados da década de 60, Zadeh observou que os recursos tecnológicos disponíveis eram incapazes de automatizar as atividades relacionadas a problemas de natureza industrial, biológica ou química, que compreendessem situações ambíguas, não passíveis de processamento através da Lógica Computacional fundamentada na Lógica Booleana. Procurando solucionar esses problemas o Prof. Zadeh publicou em 1965 um artigo (Zadeh, 1965b) resumindo os conceitos dos *Conjuntos Fuzzy*, revolucionando o assunto com a criação de *Sistemas Fuzzy*.

Em 1974, o Prof. Mahmood Mamdani (Mamdani, 1974), do Queen Mary College, Universidade de Londres, após inúmeras tentativas frustradas em controlar uma máquina a vapor com tipos distintos de controladores, incluindo o PID, somente conseguiu fazê-lo através da aplicação do raciocínio Fuzzy. Esse sucesso serviu de alavanca para muitas outras aplicações, como em 1980, no *Controle Fuzzy* de operação de um forno de cimento. Vieram em seguida, várias outras aplicações, destacando-se, por exemplo, os *Controladores Fuzzy* de plantas nucleares, refinarias, processos biológicos e químicos, trocador de calor, máquina diesel, tratamento de água e sistema de operação automática de trens.

Estimulados pelo desenvolvimento e pelas enormes possibilidades práticas de aplicações que se apresentaram, os estudos sobre *Sistemas Fuzzy* e controle de processos avançam rapidamente, culminando com a criação em 1984, da Sociedade Internacional de *Sistemas Fuzzy*, constituída, principalmente, por pesquisadores dos países mais avançados tecnologicamente.

Sistemas Fuzzy foram amplamente ignorados nos Estados Unidos porque foram associados com inteligência artificial, um campo que periodicamente se obscurecia, resultando numa falta de credibilidade por parte da indústria. A propósito disto, e apenas a título de ilustração, mais de 30% dos artigos até hoje publicados são de origem japonesa.

Os japoneses não tiveram este prejuízo; o interesse em *Sistemas Fuzzy* foi demonstrado por Seiji Yasunobu e Soji Miyamoto da Hitachi (Yasunobu, 2002), que em 1985 apresentou simulações que demonstraram a superioridade de Sistemas de Controle Fuzzy para a estrada

de ferro de Sendai; suas idéias foram adotadas e Sistemas Fuzzy foram usados para controle de aceleração, frenagem, e parada quando a linha foi inaugurada em 1987.

Outro evento em 1987 ajudou a promover o interesse em Sistemas Fuzzy: durante um encontro internacional de pesquisadores de Fuzzy em Tokyo naquele ano, Takeshi Yamakawa demostrou o uso de *Controle Fuzzy* (através de um conjunto de simples chips fuzzy dedicados) em um experimento de um pêndulo invertido – um problema clássico de controle em que um veículo tenta manter um poste montado no seu topo por uma dobradiça vertical com movimentos de ida e volta (Yamakawa, 1987).

Observadores ficaram impressionados com esta demonstração, como também com os experimentos de Yamakawa em que ele montou um copo contendo água ou até mesmo um rato vivo no topo de pêndulo; o sistema manteve estabilidade em ambos os casos. Yamakawa eventualmente foi organizando seu próprio laboratório de pesquisas de Sistemas Fuzzy ajudando a explorar suas patentes no campo.

Seguindo semelhantes demonstrações, os japoneses se apaixonaram com sistemas fuzzy, desenvolvendo tanto aplicações industriais como aplicações para consumo; em 1988 eles criaram o Laboratório Internacional de Engenharia Fuzzy (LIFE), uma cooperativa que compreendia 48 companhias para pesquisa em Sistemas Fuzzy.

Bens de consumo japoneses incorporam extensamente *Sistemas Fuzzy*. Trabalhos em *Sistemas Fuzzy* é também um procedimento nos EUA e Europa, entretanto não com o mesmo entusiasmo visto no Japão.

Em 1995, Maytag introduziu uma máquina de lavar pratos "inteligente" baseado em Controlador Fuzzy e um "one-stop sensing module" que combina um termistor (para medida da temperatura), um sensor condutivo (para medir o nível de detergente através dos íons presentes na água), um sensor de turvação que transmite luz e faz a medida da luz difundida para medir a sujeira na lavagem, e um sensor magnético para ler a taxa de giro. O sistema determina uma otimização no ciclo de lavagem, para qualquer carga obter os melhores resultados com o mínimo de energia, detergente, e água.

Pesquisa e desenvolvimento estão em andamento para aplicações Fuzzy em projeto de software, incluindo Sistemas *Fuzzy Expert* e **Integração de Lógica Fuzzy com Redes Neurais**, outros algoritmos, com o objetivo de construção de um Sistema Fuzzy capaz de aprender e imitar de forma melhor o comportamento humano.

O desenvolvimento de técnicas de Inteligência Artificial (IA), nos últimos anos, ocupa cada vez mais posição de destaque em pesquisas na área de controle de processos industriais e, aos poucos, começam a ser implantadas em plantas industriais com enorme sucesso. Dentre as técnicas mais utilizadas, além do Controle Fuzzy, podem-se destacar as Redes Neurais Artificiais aplicadas a sistemas de controle, que estão atualmente em tamanha evidência que os japoneses as consideram como duas das mais promissoras técnicas para o século XXI.

A *Lógica Difusa* ou *Lógica Fuzzy* é uma extensão da lógica booleana que admite valores lógicos intermediários entre o FALSO (0) e o VERDADEIRO (1); por exemplo o valor médio 'TALVEZ' (0,5). Isto significa que um valor lógico difuso é um valor qualquer no intervalo de valores entre 0 e 1. Este tipo de lógica engloba de certa forma conceitos estatísticos, principalmente na área de inferência.

As implementações da lógica difusa permitem que estados indeterminados possam ser tratados por dispositivos de controle. Desse modo, é possível avaliar conceitos não-quantificáveis de casos práticos tais como: avaliar a velocidade (*alta*, *média*, *baixa*, etc...), o deslocamento (*grande*, *médio*, *pequeno*...etc) e, diferentemente da lógica Booleana com valores Falso ou Verdadeiro, a veracidade de um argumento (*corretíssimo*, *correto*, *incoerente*, *totalmente erroneo*, etc..).

A Lógica Fuzzy deve ser vista mais como uma área de pesquisa sobre o tratamento da incerteza, ou uma família de modelos matemáticos dedicados ao tratamento da incerteza, do que uma *lógica* propriamente dita. A *Lógica Difusa* ou *Nebulosa*, normalmente está associada ao uso da teoria de *Conjuntos Fuzzy* proposto por Lukasiewicz sendo comum chamar a Lógica Booleana de Lógica Nítida. Muitos pesquisadores de versões booleanas de lógica não aceitam a Lógica Fuzzy como uma lógica verdadeira, no sentido em que aceitam, por exemplo, a lógica modal. Lógica Modal se refere a qualquer sistema de lógica formal que procure lidar com *modalidades* (tratar de modos quanto a tempo, possibilidade, probabilidade, etc.). Tradicionalmente, as modalidades mais comuns são *possibilidade* e *necessidade*. Lógicas para lidar com outros termos relacionados, como *probabilidade*, *eventualidade*, *padronização*, *poder*, *poderia*, *deve*, são por extensão também chamadas de lógicas modais, já que elas podem ser tratadas de maneira similar.

Uma Lógica Modal formal representa modalidades usando *operadores modais*. Por exemplo, "Era possível o impedimento de Luís Inácio" e "Luís Inácio foi possivelmente impedido" são exemplos que contêm a noção de possibilidade. Formalmente, essa noção é tratada como o operador modal *Possível*, aplicado à sentença "Luís Inácio foi impedido". Isso pode ser associado a diferentes fatos, entre eles o fato de muitos modelos permitirem soluções aproximadas que não correspondem a uma “verdade lógica”.[Fonte:Wikipédia]

Noções básicas

Na lógica proposicional, a cada proposição *p* associamos um entre dois valores possíveis: verdadeiro ou falso. É comum que sejam escolhidos valores numéricos como 1 para representar o verdadeiro e 0 para representar o falso.

Um modelo *fuzzy* simples é construído associando-se um valor $\mu(p)$ a uma proposição *p*, indicando o grau de veracidade dessa proposição, sendo que $\mu(x)$ é uma função (arbitrária) cujo conjunto imagem está entre 0 e 1 (ou 0% e 100%). Se exige pouco dessa funcional: caso *p* seja verdade, deve estar associado ao valor 100%, caso *p* seja falso deve ser associado ao valor 0%. Dessa forma, a lógica estende a lógica booleana, pois ao invés de permitir só dois valores (1 e 0) permite uma gama infinita de valores.

Da mesma forma que são estendidos os valores possíveis das proposições, também devem ser estendidos os operadores, como NÃO, E e OU. Porém, ao estender esses operadores, devemos manter certas propriedades, entre elas a compatibilidade com a versão booleana da lógica. Assim, um operador NÃO-fuzzy, ao ser aplicado sobre o valor de uma proposição fuzzy que seja 0 ou 1, deve devolver o mesmo valor que um operador NÃO retornaria na lógica booleana.

Existem uma ampla gama de funções que podem ser utilizadas como NÃO-fuzzy, E-fuzzy e OU-fuzzy, tendo sido aplicadas a vários sistemas, porém as que contém mais propriedades desejáveis e que simultaneamente são bastante fáceis de utilizar são:

NÃO-fuzzy(x) = 1 – x;
E-fuzzy(x,y) = Mínimo(x,y);
OU-fuzzy(x,y) = Máximo(x,y).

Utilizando esse modelo, podemos construir o seguinte exemplo:

Suponha que desejássemos representar de forma *fuzzy* a altura do teto de quatro carros: Alfa (1,65 m), BMW (1,75 m), Chrysler(2,0m) e Dodge(1,45 m). Nossas proposições serão da forma "X é alto", e serão:

A = Alfa é alta, $\mu(A) = 0.55$
B = BMW é alto, $\mu(B) = 0.75$
C = Chrysler é alto, $\mu(C) = 1$
D = Dodge é alto, $\mu(D) = 0$

Usando os operadores acima descritos, podemos escrever sentenças como:

Chrysler não é alto, NÃO(C), μ(NÃO(C)) = 1-μ(C) = 0
BMW não é alto, NÃO(B), μ(NÃO(B)) = 1-μ(B) = 0.25
Dodge é alto e Alfa é Alta, D e A, μ(D e A) = mínimo(μ(D),μ(A)) = 0

A lógica está claramente associada a teoria dos conjuntos. Cada afirmação (do tipo "Chrysler é alto") representa na verdade o grau de pertinência de Chrysler ao conjunto de carros altos. Isso permite que conjuntos como "alto" e "baixo" sejam tratados de forma separadas e afirmações como "Chrysler é alto 0.75" e "Chrysler é baixo 0.50" sejam válidas simultaneamente, ao contrário do que seria esperado em um modelo booleano. Esse tipo de afirmação é facilmente encontrada na descrição, por humanos, na forma como entendem certo conceito, e a Lógica Fuzzy ou difusa é uma ótima forma de tratar essa forma de incerteza.

Inferência difusa

Fazer uma inferência difusa significa aplicar regras do tipo **Se** X **Então** Y de forma que X e Y, e a própria sentença, sejam noções difusas.

Dessa forma, se torna mais fácil interpretar matematicamente e implementar sistemas a partir do conhecimento humano, como em: **Se** a temperatura é alta **E** a pressão é alta **Então** o fluxo de combustível deve ser pequeno.

É importante notar que no caso acima, uma versão de uso corrente da lógica difusa, a regra é igual a uma regra nítida que seria usada em um sistema especialista. Porém, os conjuntos (ALTO, MÉDIO e BAIXO para temperatura, por exemplo) permitem graus de pertinência, onde uma temperatura pode ter algum grau em todos os conjuntos, enquanto em um sistema exatamente nítido, apenas um valor seria possível.

Assim, em sistemas difusos, com um conjunto de regras, várias regras aparentemente contraditórias são válidas simultaneamente, possuindo ainda um grau de validade. A solução final é obtida por meio da agregação dos resultados por meio de alguma operação matemática, como o cálculo do centro de massa (Centróide) da resposta obtida.

No caso da inferência, para cada conjunto de operações básicas NÃO, E e OU escolhidos, são possíveis várias versões da implicação. Isso porque, nesta lógica, $A \rightarrow B$ (A implica B) é equivalente a várias sentenças.

Outra forma de inferência difusa é aplicar regras como o “modus ponens” (em Latim: *modo de afirmar*) é um dos modos dos silogismos condicionais e “modus tollens” (Latim: *modo que nega*) ou *negação do consequente*, é o nome formal para a *prova indireta*. Isso permite várias variações. Em uma delas, sabendo que “A implica B” de forma nítida, e tendo apenas um valor difuso de A, é possível calcular o valor de B.[Fonte Wikipédia]

Conjuntos difusos

Normalmente, o uso da lógica difusa está associado ao uso de conjuntos nebulosos.

Um conjunto nebuloso estende o conceito de conjunto permitindo que um elemento passa a ter um grau de pertinência variando entre 0 e 1, ao invés de pertencer ou não ao conjunto como na teoria de conjuntos tradicional.

Veja que o princípio é o mesmo aplicado a lógica difusa, onde o grau de veracidade pode passar a variar entre 0 e 1.

Para cada conjunto, então, é criada uma função de pertinência, que indica o grau de pertinência de seus elementos. Normalmente, essa função é criada de forma a representar algum conceito impreciso como "bem alto".

Raciocínio fuzzy

O raciocínio fuzzy também conhecido como raciocínio aproximado e pode ser dividido em 5 etapas:

Transformação das variáveis do problema em valores fuzzy, ou fuzzificação
Aplicação dos operadores fuzzy
Aplicação da implicação
Combinação de todas as saídas fuzzy possíveis
Transformação do resultado fuzzy em um resultado nítido, a defuzzificação.

No primeiro passo, para cada valor de entrada associamos uma função de pertinência, que permite obter o *grau de verdade* da proposição:

Determinar o grau de pertinência de cada conjunto (proposição);
Limitar o valor da entrada entre 0 e 1;

O segundo passo é aplicar os operadores fuzzy, assim como os operadores da lógica booleana. Os operadores usados na Lógica Fuzzy são 'AND' e 'OR', conhecidos como operadores de relação. Na Lógica Fuzzy são utilizados para definir o grau máximo e mínimo de pertinência do conjunto.

O terceiro passo é aplicar o operador de implicação, usado para definir o peso no resultado e remodelar a função, ou seja, o terceiro consiste em criar a hipótese de implicação. Como no exemplo abaixo:

O motor é potente OU o consumo é elevado ENTÃO abastecimento é alto.

No quarto passo ocorre a combinação de todas as saídas em um único conjunto fuzzy, algo semelhante ao processo de união e intersecção, na teoria dos conjuntos abruptos.

O quinto e último passo no processo do raciocínio fuzzy, é a 'defuzzyficação' que consiste em retornar os valores, obter um valor numérico dentro da faixa estipulada pela Lógica Fuzzy.

Um exemplo simples que demonstra o processo de pertinência do raciocínio fuzzy seria:

Se A é identificado como 'o tanque está vazio' e B como 'o combustível acabou', então se é verdade que 'o tanque está vazio', é também verdade que 'o combustível acabou'.

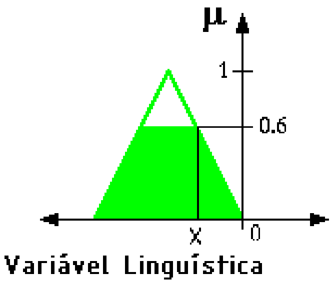
Essa seria um exemplo pensado na lógica tradicional onde:

Fato: x é A;
Regra: se x é A então y é B;
Conclusão: y é B.

Esta regra aplica um conceito aproximado. Porem se pensarmos desta forma: se nós temos a mesma regra de implicação SE *o tanque está vazio*, ENTÃO *o combustível acabou'* e nós sabemos que o tanque está mais ou menos vazio, então nós podemos inferir que o combustível está mais ou menos quase acabando. Ou seja:

Fato: x é A' (quase A);
Regra: se x é A então y é B;
Conclusão: y é B' (quase B).

Este conceito de fuzzyficação funciona da seguinte forma se A' está próximo de A (situação inicial) e B' está próximo de B (inicial). A, A', B e B' fazem parte do conjunto universo, chegando assim ao paradigma do raciocínio *fuzzyano*, também chamado de *modus*



ponens generalizado.

Figura. A3.1.Correlação entre um valor X de um universo de discurso com a sua pertinência μ.

Modelamento por Lógica Nebulosa (LN ou Lógica Fuzzy)

A teoria dos conjuntos nebulosos se baseia na suposição de que os elementos chaves do pensamento humano não são objetos que possuam uma classificação exata, mas sim um conjunto de classes de objetos que são conjuntos com fronteiras difusas (daí o nome), ou seja, a transição de classes é gradual ao invés de abrupta.

O conceito de fuzzificação é introduzido, quando se define qual o grau de pertinência de um valor x de uma variável nebulosa X de uma função característica μ(x) de modo que ela possa assumir um número infinito de valores diferentes no intervalo de pertinência [0,1] [Bib. 3.3.1].

Treinamento em rede neural

Sintaxe

[rnavoz,tr] = train(RNAVOZ,P,T,Pi,Ai)
[rnavoz,tr] = train(RNAVOZ,P,T,Pi,Ai,VV,TV)

Descrição

TRAIN Treina uma rede RNAVOZ de acordo com RNAVOZ.trainFcn e RNAVOZ.trainParam.
TRAIN(RNAVOZ,P,T,Pi,Ai) onde,
RNAVOZ - Rede.
P - Parâmetros de entrada de Rede.
T - Objetivos de Rede, default = zeros.
Pi - condições iniciais de atraso de entradas, default = zeros.
Ai - condições iniciais de atraso de camada, default = zeros.

e retorna:
rnavoz - Nova rede.
tr - Registro do Treinamento (época e performance).

Notar que T é opcional e só precisa que seja usado para redes que requerem objetivos. Pi e Ai também são opcionais e necessitam que só sejam usados para redes que têm atrasos de entrada ou camada.

Os argumentos de sinal de TRAIN podem ter dois formatos: arranjo de célula ou matriz.

O formato de arranjo de célula é mais fácil descrever. Ele é mais conveniente para redes com múltiplas entradas e saídas, e permite apresentar sequencias de entradas:

P - NixTS arranjo de célula, cada elemento P{i,ts} é uma matriz de RixQ.
T - NtxTS arranjo de célula, cada elemento P{i,ts} é uma matriz de VixQ.
Pi - NixID arranjo de célula, cada elemento Pi{i,k} é uma matriz de RixQ.
Ai - NlxLD arranjo de célula, cada elemento Ai{i,k} é uma matriz de SixQ.

Onde:
Ni = rnavoz.numInputs
Nl = rnavoz.numLayers
Nt = rnavoz.numTargets
ID = rnavoz.numInputDelays
LD = rnavoz.numLayerDelays
TS = número de passos de tempo
Q = tamanho de grupo
Ri = rnavoz.inputs{i} .size
Si = rnavoz.layers{i} .size
Vi = rnavoz.targets{i} .size

As colunas de Pi e Ai são ordenadas da condição do atraso mais velho para o mais recente:

Pi{i,k} = entrada i no tempo ts=k-ID.
Ai{i,k} = saída de camada i no tempo ts=k-LD.

O formato de matriz pode ser usado se só uma vez de tempo for ser simulado (TS = 1). É conveniente para redes com sòmente uma entrada e saída, mas pode ser usado com redes que tenham mais.

Cada argumento de matriz é achado armazenando os elementos do argumento de arranjo de célula correspondente em uma única matriz:

P - (soma de matriz de Ri)xQ
T - (soma de matriz de Vi)xQ
Pi - (soma de Ri)x(ID*Q) matriz.
Ai - (soma de Si)x(LD*Q) matriz.

TRAIN(RNAVOZ,P,T,Pi,Ai,VV,TV) possui estruturas opcionais de validação e vetores de teste:

VV.P, TV.P - Entradas de Validação/teste.
VV.T, TV.T - Objetivos de Validação/teste, default = zeros.
VV.Pi, TV.Pi - Validação/teste de condições iniciais de atraso de entradas, default = zeros.
VV.Ai, TV.Ai - Validação/teste de condições de atraso de camada, default = zeros.

Os vetores de validação são usados para deixar de treinar mais cedo se mais adiante treinamento nos vetores primários ferirem a generalização para os vetores de validação. Teste de desempenho de vetor pode ser usado para medir quão bem a rede generaliza além dos vetores primário e de validação.

Se são fixados VV.T, VV.Pi, ou VV.Ai a uma matriz vazia ou arranjo de célula, serão usados valores “default”. O mesmo é verdade para VT.T, VT.Pi, VT.Ai.

Exemplos:

Aqui entradas P e objetivos T definem uma função simples que nós podemos delinear:

```
p = [0 1 2 3 4 5 6 7 8];
t = [0 0.84 0.91 0.14 -0.77 -0.96 -0.28 0.66 0.99];
plot(p,t, ' o')
```

Aqui NEWFF é usado para criar uma rede de duas camadas com alimentação adiante.

A rede terá uma entrada (variando de 0 a 8), seguida por uma camada de 10 neurônios TANSIG, seguida por uma camada com 1 Neurônio PURELIN. A retropropagação TRAINLM (treinamento por Levenberg/Marquardt) é usada. A rede também é simulada.

```
rede = newff([0 8],[10 1], {' tansig' ' purelin'}, ' trainlm');
y1 = sim(rnavoz,p)
plot(p,t, ' o',p,y1, ' x')
```

Aqui a rede é treinada para até 50 épocas para uma meta de erro de 0.01, e então resimulada.

```
rnavoz.trainParam.epochs = 50;
rnavoz.trainParam.goal = 0.01;
rede = train(rnavoz,p,t);
y2 = sim(rnavoz,p)
plot(p,t, ' o',p,y1, ' x',p,y2, ' * ')
```

Algoritmo

A função TRAIN chama a estrutura indicada por rnavoz.trainFcn, enquanto usa os

valores de parâmetro de adaptação indicados por rnavoz.trainParam.

Tipicamente uma época de treinamento está definida como uma única apresentação de todos os vetores de entrada para a rede. A rede é atualizada então de acordo com os resultados de todas essas apresentações.

O treinamento acontece até que um número de máximo de épocas acontece, a meta de desempenho é encontrada, ou qualquer outra condição de parada da função rnavoz.trainFcn acontece.

Algumas funções de treinamento partem desta norma só apresentando um vetor de entrada (ou sequencia) cada época. Um vetor de entrada (ou sequencia) é fortuitamente escolhido a cada época de vetores de contribuição concorrentes (ou sucessivos) de entrada.

NEWC e NEWSOM devolvem redes que usam TRAINWB1, uma função de treinamento, que faz isto.

TRAINBFG o BFGS quasi-Newton backpropagation (retropropagação).

Sintaxe:

[rnavoz,tr] = trainbfg(rnavoz,Pd,Tl,Ai,Q,TS,VV)
info = trainbfg(code)

Descrição

TRAINBFG é uma rede que treina função que atualiza peso e valores de polarização de acordo com o método BFGS quasi-Newton .

TRAINBFG(RNAVOZ,Pd,Tl,Ai,Q,TS,VV,TV) pega estas entradas, onde:

RNAVOZ - rede Neural.
Pd - vetores de entrada atrasados.
Tl - vetores designados de camada.
Ai - condições de atraso de entradas iniciais.
Q - tamanho de Grupo.
TS - passos de Tempo.
VV - Qualquer matriz vazia [ ] ou estrutura de vetores de validação.
TV - Qualquer matriz vazia [ ] ou estrutura de vetores de teste.
e retorna,
rnavoz - rede treinada .
TR - registro de treinamento de vários valores em cima de cada época:
TR.epoch - número de Época.
TR.perf - desempenho de Treinamento .
TR.vperf - desempenho de Validação.
TR.tperf - desempenho de Teste.

O treinamento acontece de acordo com os parâmetros que TRAINBFG está treinando, mostrado aqui com os seus valores default:

rnavoz.trainParam.epochs 100	Número máximo de épocas para treinar
rnavoz.trainParam.show 25	Épocas entre progresso mostrado
rnavoz.trainParam.goal 0	Meta de desempenho
rnavoz.trainParam.time_inf	Máximo tempo para treinar em segundos
rnavoz.trainParam.min_grad 1e-6	Mínimo gradiente de desempenho

rnavoz.trainParam.max_fail 5	Máximos fracassos de validação
rnavoz.trainParam.searchFcn 'srchcha'	Nome de rotina de procura de linha para usar.

Parâmetros relacionados para métodos de procura de linha (nem todos usados para todos os métodos):

rnavoz.trainParam.scal_tol 20	Dividem em delta para determinar tolerância por procura linear.
rnavoz.trainParam.alpha 0.001	Fator de escala que determina redução suficiente em perf.
rnavoz.trainParam.beta 0.1	Fator de escala que determina tamanho de passo suficientemente grande.
rnavoz.trainParam.delta 0.01	Tamanho de passo inicial em intervalo local .
rnavoz.trainParam.gama 0.1	Parâmetro para evitar reduções pequenas em desempenho normalmente estabelecido para 0.1. (Veja uso em SRCH_CHA.)
rnavoz.trainParam.low_lim 0.1	Baixo limite menor em mudança em tamanho de passo.
rnavoz.trainParam.up_lim 0.5	Limite Superior em mudança em tamanho de passo.
rnavoz.trainParam.maxstep 100	Comprimento de passo Máximo.
rnavoz.trainParam.minstep 1.0e-6	Comprimento de passo Mínimo.
rnavoz.trainParam.bmax 26	Tamanho de passo Máximo.

As dimensões para estas variáveis são:

Pd - NoxNixTS arranjo de célula, cada elemento P{i,j,ts} é uma matriz de DijxQ.
Tl - NlxTS arranjo de célula, cada elemento P{i,ts} é uma matriz de VixQ.
Ai - NlxLD arranjo de célula, cada elemento Ai{i,k} é uma matriz de SixQ.

Onde:
Ni = rnavoz.numInputs
Nl = rnavoz.numLayers
LD = rnavoz.numLayerdelays
Ri = rnavoz.inputs{i}.size
Si = rnavoz.layers{i}.size
Vi = rnavoz.targets{i}.size
Dij = Ri * length(rnavoz.inputWeights{i,j}.delays)

Se VV não é uma matriz vazia [ ], deve ser uma estrutura de vetores de validação, a seguir:

VV.PD - entradas de validação atrasadas.
VV.Tl - objetivos de validação de camada .
VV.Ai - condições de validação de entradas iniciais.
VV.Q - tamanho de grupo de validação.
VV.TS - passos de tempo de validação.

que é usada para deixar de treinar mais cedo se o desempenho de rede nos vetores de validação não melhora ou permanece o mesmo para épocas de MAX_FAIL seguidos.

Se a TV não é uma matriz vazia [ ], deve ser uma estrutura de vetores de validação,

TV.PD - entradas de validação atrasadas.
TV.Tl - objetivos de validação de camada .
TV.Ai - condições de validação de contribuição iniciais.
TV.Q - tamanho de grupo de Validação.
TV.TS - passos de tempo de Validação.

que é usada para testar a capacidade de generalização da rede treinada.

TRAINBFG(CODE) retorna informação útil para cada sentença de CÓDIGO:

' pnames'   - Nomes de parâmetros de treinamento.
' pdefaults' - parâmetros de treinamento default.

Uso de rede

Você pode criar uma rede "standard" que usa TRAINBFG com:

NEWFF, NEWCF, ou NEWELM.

Preparar uma rede de usuário customizada treinada com TRAINBFG :

Estabelecer rnavoz.trainFcn para ' trainbfg'.

Este estabelecerá rnavoz.trainParam aos parâmetros default de TRAINBFG.
2) Estabeleça as propriedades de rnavoz.trainParam para valores desejados.

Em qualquer caso, chamando TRAIN com a rede resultante, vai treinar a rede com TRAINBFG.

Exemplos:

Aqui está um problema consistindo em parâmetros de entrada P (nas equações literárias são denominados de x) e alvo T (valor y desejado na saída) que nós gostaríamos de resolver com uma rede.

```
p = [0 1 2 3 4 5];
t  = [0 0 0 1 1 1];
```

Aqui, uma rede de duas-camadas com alimentação-adiante (feedforward) é criada.
As entradas da rede variam de [0 a 10]. A primeira camada tem dois neurônios TANSIG, e a segunda camada tem um neurônio LOGSIG. A função que treina a rede TRAINBFG será usada.

```
% Crie e Teste uma Rede

rnavoz = newff([0 5],[2 1], {' tansig', ' logsig'}, ' trainbfg');
a = sim(rnavoz,P)
```

```
% Treina e Retesta a Rede

rnavoz.trainParam.epochs = 50;
rnavoz.trainParam.show = 10;
rnavoz.trainParam.goal = 0.1;
rnavoz = train(rnavoz,P,T);
a = sim(rnavoz,P)
```

Algoritmo

TRAINBFG pode treinar qualquer rede contanto que seu peso, contribuição líquida, e funções de transferência têm funções derivadas.

A retropropagação (Backpropagation) é usada para calcular a derivada de desempenho PERF com respeito ao peso e variáveis de pré-conceito X. Cada variável é ajustada de acordo com a seguinte:

X = X + a*dX;

Onde:

dX é a direção de procura. O parâmetro a é selecionado minimizar o desempenho ao longo da direção de procura.

A função de procura de linha searchFcn é usada para localizar o ponto mínimo.

A primeira direção de procura é o negativo do gradiente de desempenho.

Em repetições sucessivas é computada a direção de procura de acordo com a fórmula seguinte:

dX = -H\gX;

Onde:

gX é o gradiente e H é uma matriz de Hesse aproximada.

Veja (Brânquia, Murray & Wright 1981) para uma discussão do método BFGS quasi-Newton mais detalhada e (Langruber 2003).

O treinamento para, quando qualquer de uma destas condições acontece:
o número de máximo de épocas (EPOCHS) ou repetições é alcançado.
a quantia de máximo de TEMPO foi excedida.
desempenho foi minimizado à META.
o gradiente de desempenho cai debaixo de MINGRAD.
desempenho de validação aumentou mais que tempos de MAX_FAIL
desde a última vez que diminuiu (ao usar validação).

Existem outras redes que podem ser treinadas de acordo com estes procedimentos:
NEWFF, NEWCF, TRAINGDM, TRAINGDA, TRAINGDX, TRAINLM,
TRAINRP, TRAINCGF, TRAINCGB, TRAINSCG, TRAINCGP, TRAINOSS.

APD.4. MODELO MATEMÁTICO E SIMULAÇÃO

Para um cilindro maciço de massa ***M*** e raio de base ***R***, em torno de seu eixo:

$$J = \frac{1}{2}MR^2 \quad \text{(eq.4.6)}$$

Para uma esfera maciça de massa ***M*** e raio ***R*** , em torno de seu centro:

$$J = \frac{2}{5}MR^2 \quad \text{(eq.4.7)}$$

Para um anel cilíndrico de massa ***M*** e raio ***R*** , em torno de um eixo paralelo à geratriz e passando por seu centro:

$$J = MR^2 \quad \text{(eq.4.8)}$$

Para um cilindro vazado de raio externo R e de raio interno ***r***, em torno do seu eixo:

$$J = \frac{M}{2}(r^2 + R^2) \quad \text{(eq.4.9)}$$

Para uma barra delgada, com área de seção transversal tendendo a **0** e comprimento d, perpendicularmente à barra e passando por seu centro:

$$J = \frac{1}{12}Md^2 \quad \text{(eq.4.10)}$$

Essa parte refere-se às coordenadas da cadeira em relação a um eixo de referência:

MW=[0.64 0.64 0.56 0.56 0.24 0.24 0.48 0.45 2.43 2.43];

Essa parte refere-se às massas da cadeira. A seguir é apresentada a parte do programa referente a primeira etapa:

```
SOMPROW=[XW*MW'YW*MW'ZW*MW'];%Soma do produto
SOMASW=sum(MW,2);%Soma das massas
CGW=[SOMPROW(1)/SOMASW
SOMPROW(2)/SOMASW
SOMPROW(3)/SOMASW];
```

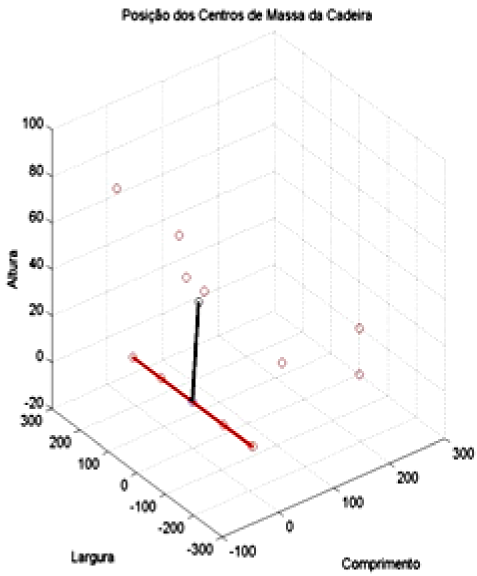


Figura A4.1 - As diversas coordenadas estáticas dos centros de massa da cadeira e o CG resultante.

É efetuado o produto das massas pelas coordenadas e o resultado é dividido pela massa total. Assim, determina-se as coordenadas do CG da cadeira.

A segunda parte do programa simula aleatoriamente as variações das forças que atuam nos quatros extensômetros, $S_{1,}$ $S_{2,}$ $S_{3,}$ $S_{4,}$ para simular a movimentação do cadeirante sentado. Em seguida, é efetuado o cálculo de CG humano. Essa parte refere-se às coordenadas de cada sensor em relação a um eixo de referência. São coordenadas dos sensores:

```
xs1=0.05; xs2=0.25; xs3=0.25; xs4=0.05;
ys1=0.10; ys2=0.10; ys3=-0.10; ys4=-0.10;
zs1=0.25; zs2=0.25; zs3=0.25; zs4=0.25;
```

Tais coordenadas são dispostas em forma de matriz:

```
Xs=[xs1 xs2 xs3 xs4];
Ys=[ys1 ys2 ys3 ys4];
Zs=[zs1 zs2 zs3 zs4];
```

A seguir é apresentada a parte do programa referente à segunda etapa:

```
Mh=randn(1,4); %Entrada randomica de forças
Xh=[0.1 0.2 0.2 0.1];
Yh=[0.03 0.03 -0.03 -0.03];
Zh=[0.35 0.35 0.35 0.35];
SOMPROh=[Xh*Mh' Yh*Mh' Zh*Mh'];
SOMASh=sum(Mh,2);
CGh=[SOMPROh(1)/SOMASh SOMPROh(2)/SOMASh SOMPROh(3)/SOMASh];
```

Nesse trecho do programa, é simulada a variação das forças aleatoriamente e, na sequência, é efetuado o produto das forças pelas coordenadas dos sensores, determinado-se assim a coordenada do CG humano.

Na terceira parte do programa é efetuada a soma dos dois resultados anteriores para a composição do CG resultante que será usado no controle da cadeira. A seguir, é apresentada a parte do programa referente a esta etapa para o calculo do CG resultante:

```
SOMPRO=SOMPROW+SOMPROh;
SOMAS=SOMASW+SOMASh;
```

CG=[SOMPRO(1)/SOMAS SOMPRO(2)/SOMAS SOMPRO(3)/SOMAS];

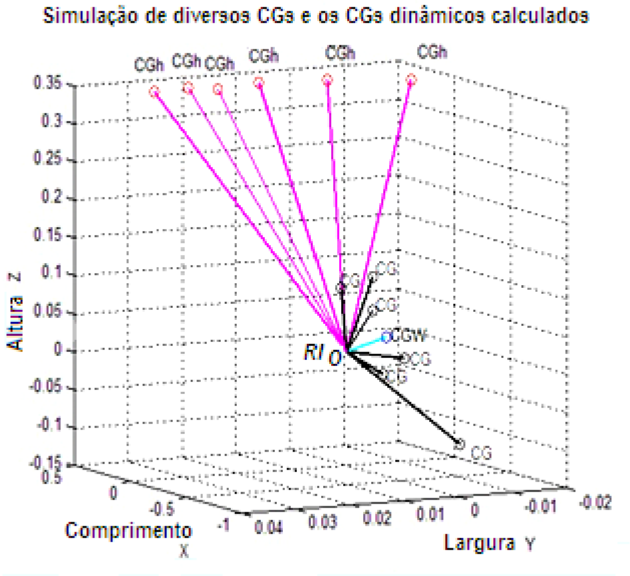


Figura A4.2 - Composição de diversos centros de massa humanos simulados (CGh) + CGw (fixo) gerando o CG resultante.

Na terceira etapa, é efetuada a soma das coordenadas dos CGs humanos e da cadeira, obtendo-se, assim, o CG resultante.

Movimentos Rotacionais

As rotações finitas são um dos aspectos da cinemática e, apesar da existência de diversas metodologias, a que foi empregada neste trabalho se baseia fundamentalmente em Newton e nos ângulos de Euler, uma das parametrizações mais encontradas na literatura, por apresentar uma visualização mais imediata do problema.

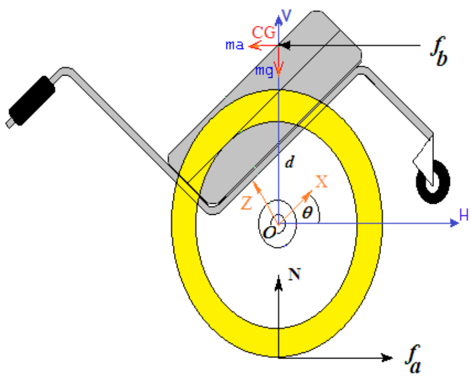
Apesar disso, essa metodologia pode levar a determinadas singularidades que devem ser evitadas ao longo do projeto, através de uma limitação dos movimentos.

No projeto dessa cadeira de rodas, as limitações podem ocorrer naturalmente, ou seja, as estruturas mecânicas nas bordas servem de batente, como nos projetos das cadeiras de rodas convencionais elétricas ou não. Mas tais limitações podem ocorrer de uma forma não natural, como é o caso do programa de controle limitar o movimento em função dos sensores.

No plano xz (Figura A4.3)

Rotação em torno do eixo y :

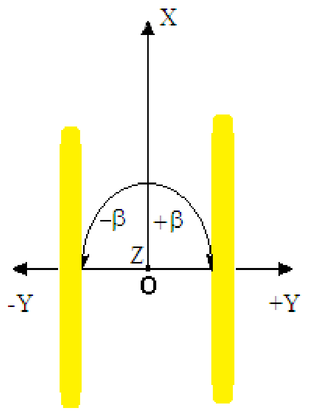
$$R_{Y\theta} = \begin{bmatrix} \cos\theta & -sen\theta & 0 \\ sen\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



No plano xy (Figura A4.4)

Rotação em torno do eixo z:

$$R_{Z,\phi} = \begin{bmatrix} \cos\beta & 0 & sen\beta \\ 0 & 1 & 0 \\ -sen\beta & 0 & \cos\beta \end{bmatrix}$$



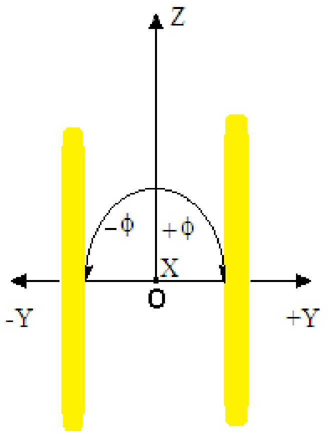
Convém lembrar que, conforme a movimentação que se faça nesse plano (em torno do eixo Z), pode, ou não, ocorrer anteriormente uma rotação no plano xy (cadeira em equilíbrio), pois, dessa forma, este movimento seria uma matriz produto das duas matrizes de rotação.

No plano zy (Figura A4.5)

A cadeira pode entrar em um plano inclinado e, ao torcer este plano lateralmente, temos que considerar o movimento de forma inevitável, conforme se vê abaixo:

Rotação em torno do eixo x :

$$R_{X,\varphi} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -sen\phi \\ 0 & sen\phi & \cos\phi \end{bmatrix}$$



Apesar de esta rotação ser possível (no plano inclinado lateral), colocou-se uma restrição inicial neste trabalho para tal movimento, de forma que as duas rodas da cadeira estejam sempre niveladas simultaneamente e essa transformação ou correção já não seria considerada.

Movimento Translacional

As combinações de movimentos nesses planos, para incluir os movimentos de translação, tornam o modelo ainda mais complexo, conforme demonstrado a seguir.

No plano XZ

A consideração da translação nesse plano faz sentido quando a cadeira avança ou recua na direção do eixo X, no plano horizontal, e/ou sobe uma rampa em linha reta, sendo que,

para este caso, o deslocamento do referencial local em relação ao inercial se faz sentir através de uma variação tanto de **x** como de **z**.

No plano YZ

A consideração nesse plano só faz sentido quando a cadeira sobe uma rampa com giro sobre esta para a esquerda ou para a direita e o deslocamento do referencial local em relação ao inercial se faz sentir tanto para y como para z. No caso deste estudo, considera-se que as rampas são planas e, mesmo no caso de giro, não existem deslizamentos que mudem sua posição.

No plano XY

Os movimentos nesse plano são mais ilimitados devido à composição que se pode fazer para atingir esses objetivos pré-determinados, só havendo restrição lateral para este tipo de translação. Considerando a Figura 4.1.7 a seguir:

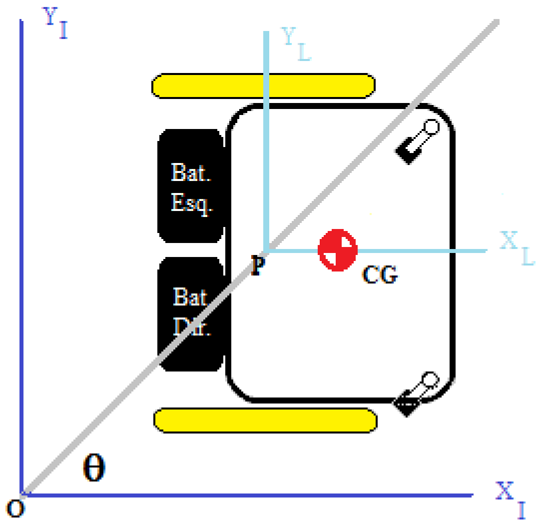


Figura A4.6 - Diagrama espacial da cadeira de rodas no plano XY.

Considera-se que:

b = largura entre a roda e ***O*** o referencial inercial, no chassis da plataforma;
V_e = a velocidade da roda esquerda = $\omega_e r_e$ (onde r_e é o raio da roda esquerda = r = constante);
V_d = a velocidade da roda direita = $\omega_d r_d$ (onde rd é o raio da roda direita = r = constante) e;
R = a distância do ponto médio **I** do eixo ao **CIR** (Centro Instantâneo de Referência).

No plano ZY:

Esse movimento não existe devido a restrição de não escorregamento lateral.

Centro de referência inercial

Conforme a Figura 4.11., a roda da cadeira é livre para rodar em torno do seu eixo (eixo z), sendo que ela se desloca preferencialmente em uma direção (eixo x). O eixo de referência vertical continua sendo o eixo y. Esse modelo, para baixas velocidades, é razoável.

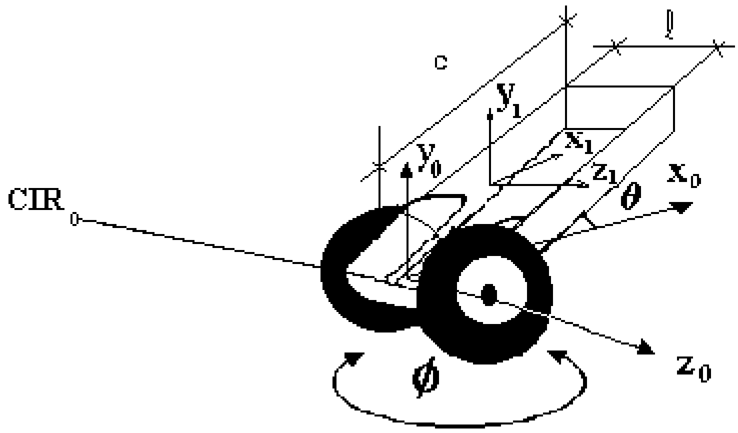


Figura A4.7 - Referenciais espaciais da cadeira de rodas

Para a cadeira de rodas exibir um movimento de giro, deve haver um ponto de referência no qual cada roda segue um curso circular. Esse ponto, denominado de centro instantâneo de rotação (CIR), muda de acordo com as mudanças na cadeira realizadas através de suas rodas em relação ao seu eixo vertical (orientação), embora mantendo sempre o alinhamento em relação ao eixo de giro da roda (eixo z). Para a consideração de tal modelo (no caso, diferencial), há que se considerar que a cadeira tem dois graus de liberdade baseados em sua posição no plano zy e o ângulo de orientação ϕ.

Diferenciais são os mecanismos mais simples nessa área de locomoção e aplicáveis no caso dessa cadeira na qual dois motores independentes acionam duas rodas, separadamente. Para a cadeira, cada uma das rodas exibe um movimento de rotação em torno de um eixo comum. Ao se variar a velocidade relativa das rodas, diferentes posicionamentos podem ser obtidos. Para cada tempo **t,** o ponto no qual a cadeira gira deve ter a propriedade de que tanto a roda direita como esquerda sigam uma trajetória em relação ao seu CIR com uma mesma velocidade angular **ω,** tal que:

$$\omega (R + b/2) = V_d \qquad \text{(eq.4.12)}$$

$$\omega (R - b/2) = V_e \qquad \text{(eq.4.13)}$$

Onde:

b é a distancia ao longo do eixo das rodas, tomando como referencia o centro destas, V_d é a velocidade da roda direita, V_e é a velocidade da roda esquerda e R é a distancia do centro do eixo das rodas até o CIR. É importante notar que, com exceção de **b** e **d**, todos os outros parâmetros são função do tempo. Resolvendo para R e ω tem-se:

$$R = d\,(V_e + V_d)/2\,(V_d - V_e) \qquad \text{(eq.4.14)}$$

$$\omega = (V_d - V_e) / b \qquad \text{(eq.4.15)}$$

Determinam-se diversos casos de interesse a seguir:

Se $V_e = V_d$, então o raio R é infinito e a cadeira move em linha reta;

Se + V_e = - V_d , ou vice – versa, então o raio R é zero e a cadeira gira sobre o ponto médio do eixo entre as rodas, sendo este o princípio adotado para que, na Cinética, a simetria causada por esta referência simplifique a modelagem, assim como elimine comportamentos dinâmicos difíceis de se estabilizar pelos controles baseados em inteligência artificial, de certa forma inovadores, que serão adotados;

Se $V_e \neq V_d$, então a cadeira deslocaria em trajetória curva em relação ao seu CIR. Nesse caso da cadeira equilibrista, considera-se sempre o giro em relação ao referencial inercial e esta possibilidade não ocorre. De qualquer forma, o estudo do CIR é mantido para variantes futuras deste trabalho.

Cinemática direta

Supondo que a cadeira está em uma posição (z,x), ao longo de uma linha, fazendo um angulo ϕ com o eixo z, conforme a Figura 4.4. . Através de manipulação dos parâmetros de controle V_e e V_d, a cadeira pode tomar diversas posições. Determinando a posição que a cadeira atingiu, s os parâmetros de controle, é o que se denomina cinemática direta e o seu CIR é dado por:

CIR = (z-Rsenϕ , x+Rcosϕ) (eq.4.16)

Pelo fato de V_e e V_d , assim como R e ω , serem funções do tempo, então em t + δt a posição da cadeira será dada por:

$$\begin{bmatrix} \dot{z} \\ \dot{x} \\ \phi \end{bmatrix} = \begin{bmatrix} \cos(\omega\delta t) & -\mathrm{sen}(\omega\delta t) & 0 \\ \mathrm{sen}(\omega\delta t) & \cos(\omega\delta t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} z - CIR_z \\ x - CIR_x \\ \phi \end{bmatrix} + \begin{bmatrix} CIR_z \\ CIR_x \\ \omega\delta t \end{bmatrix}$$

A expressão anterior determina a posição da cadeira no tempo t, desde que se conheça V_e e V_d e integrando, conforme abaixo. Para o caso das rodas serem diferenciais, tem-se:

$z(t) = \frac{1}{2} \int_o^t (V_d(t) + V_e(t)) \cos(\phi(t))\, dt$ (eq.4.17)

$x(t) = \frac{1}{2} \int_o^t (V_d(t) + V_e(t))\, \mathrm{sen}(\phi(t))\, dt$ (eq.4.18)

$\phi(t) = 1/l \int_o^t (V_d(t) - V_e(t))\, dt$ (eq.4.19)

Cinemática inversa

Para determinar os parâmetros da cadeira, para atingir um ponto desejado e conhecido no espaço, tem-se um problema mais delicado denominado cinemática reversa ou inversa. As equações anteriores, 3.2, descrevem uma restrição na velocidade da cadeira que não pode ser integrada para ocasionar uma restrição de posicionamento. Isso é conhecido como uma

restrição –não holonômica, cuja solução, em geral, é difícil.Contudo, para determinadas classes de funções de controle $V_e(t)$ e $V_d(t)$, algumas soluções são possíveis. Se se assumir que $V_e(t) = V_e$ e $V_d(t) = V_d$ e, ainda, $V_e \neq V_{d,}$ isso permite que:

$z(t) = (b/2)\ (\ (V_d + V_e) / (V_d - V_e)\ )sen((t/b)\ (V_d - V_e)$ (eq.4.20)

$x(t) = -\ (b/2)\ (\ (V_d + V_e) / (V_d - V_e)\ )cos((t/b)\ (V_d - V_e) + (l/2)\ (\ (V_d + V_e) / (V_d - V_e)\ )$

(eq.4.21)

$\phi(t) = t/b(V_d - V_e)\ dt$ (eq.4.22)

Onde:
$(z,x,\phi)_{t\,=\,0} = (0,0,0)$.

Dado um tempo a ser atingido t e uma posição conhecida (z,x), as equações anteriores resolvem para V_d e V_e , mas não para um controle independente de ϕ. Para isso, é necessário que a junção dos movimentos rotacionais com os translacionais formem a dinâmica completa do futuro modelo matemático da cadeira.

Obs: Existem infinitas soluções para V_d e V_e, considerando essas equações, que levam à solução do problema, sendo que a cadeira produz trajetórias curvas em diferentes números de tempo e em diferentes direções.

Melhor que tentar inverter as equações 5.2.x, para determinar os parâmetros de controle que permitam uma determinada posição, considera-se dois casos especiais de movimento:

Se $V_d = V_e = V$, os movimentos da cadeira simplificam para:

$$\begin{bmatrix} \dot{z} \\ \dot{x} \\ \phi \end{bmatrix} = \begin{bmatrix} z + v\cos(\phi)\delta t \\ x + v\,\mathrm{sen}(\phi)\delta t \\ \phi \end{bmatrix}$$

Então, a cadeira move em linha reta.

Se em módulo, $V_d = -V_e = V$, os movimentos da cadeira simplificam para:

Então, a cadeira gira em torno do eixo y.

$$\begin{bmatrix} \dot{z} \\ \dot{x} \\ \phi \end{bmatrix} = \begin{bmatrix} z \\ x \\ \phi + 2v\delta t / l \end{bmatrix}$$

O princípio do Pêndulo Invertido

O carro de massa **M** desliza sem atrito no eixo dos x. Um pêndulo de massa m e de comprimento d está fixo ao carro. Aplica-se uma força **F** ao carro de maneira a manter o pêndulo por cima do carro, segundo as notações:

Pêndulo: $d\theta/dt = \theta'$ e $d^2\theta/dt^2 = \theta''$ Carro: $dx/dt = x'$ e $d^2x/dt^2 = x''$

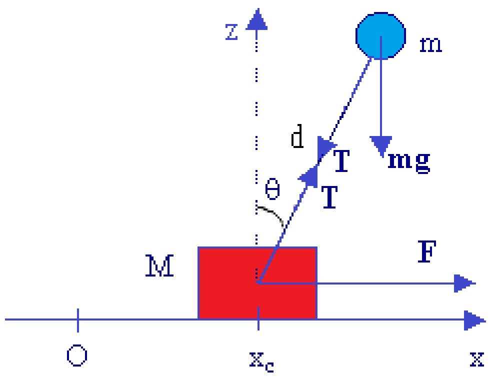


Figura 4.12. – Diagrama geométrico para modelagem do pêndulo na inclinação.

Equações diferenciais do movimento

Aproximação Newtoniana

Sendo $\mathbf{a_r}$ a aceleração relativamente ao eixo do pêndulo e $\mathbf{a_c}$, a aceleração do eixo, tem-se:

$m(\mathbf{a_r} + \mathbf{a_c}) = \mathbf{T} + Mg$ (eq.4.23)

Projeta-se o comprimento do pêndulo:

$m(d\theta'^2 - x'' \operatorname{sen}\theta) = T + mg\cos\theta$

Logo:

$\mathbf{T = m\,d\theta'^2 - m\,x''\operatorname{sen}\theta - mg\cos\theta}$ **(Relação 1)**

$J\,d^2\theta/dt^2 = M_{mg} + M_T - M_{mac}$

$m\,d^2\theta'' = mgd\operatorname{sen}\theta + 0 - mx''\cos\theta\, d$

$\mathbf{d\theta'' = g\operatorname{sen}\theta - x''\cos\theta}$ **(Relação 2)**

Equação de Newton para o carro

$M\,\mathbf{a} = \mathbf{T} + \mathbf{F} + M\mathbf{g} + \mathbf{R_N} = \mathbf{T} + \mathbf{F}$

Projeta-se o comprimento do pêndulo:

$M\,x'' = T\operatorname{sen}\theta + F$

Substitui-se **T** pelo seu valor (Relação 1), obtendo-se:

$M\,x'' = m\,d\theta'^2\operatorname{sen}\theta - m\,x''\operatorname{sen}^2\theta - mg\cos\theta\operatorname{sen}\theta + F$

Segundo **(2)**,

$mg\operatorname{sen}\theta = m\,d\theta'' + m\,x''\cos\theta$

Logo:

$M\,x'' = m\,d\theta'^2\operatorname{sen}\theta - m\,x''\operatorname{sen}^2\theta - m\,d\theta''\cos\theta - m\,x''\cos^2\theta + F$

$\quad = m\,d\theta'^2\operatorname{sen}\theta - m\,x'' - m\,d\theta''\cos\theta + F$

Onde:

$$(M + m)\, x'' = m\, d\theta'^2 \text{sen}\theta - m\, d\theta'' \cos\theta + F \qquad \textbf{(Relação 3)}$$

Substitui-se **x''** na relação **2**:

$$(M + m)\, d\theta'' = (M + m)\, g\, \text{sen}\theta - m\, d\theta'^2 \text{sen}\theta \cos\theta + m\, d\theta'' \cos^2\theta - F \cos\theta$$

$$(M + m\, \text{sen}^2\theta)\, d\theta'' = (M + m)\, g\, \text{sen}\theta - m\, d\theta'^2 \text{sen}\theta \cos\theta - F \cos\theta$$

$$d\theta'' = ((M + m)\, g\, \text{sen}\theta - m\, d\theta'^2 \text{sen}\theta \cos\theta - F \cos\theta)/(M + m\, \text{sen}^2\theta)$$

Substitui-se dθ'' na relação 3:

$$(M + m)\, x'' = m\, d\theta'^2 \text{sen}\theta - m\, ((M + m)\, g\, \text{sen}\theta \cos\theta - m\, d\theta'^2 \text{sen}\theta \cos^2\theta - F \cos^2\theta)/(M + m\, \text{sen}^2\theta) + F$$

$$(M + m)\, x'' = m\, d\theta'^2 \text{sen}\theta\, (M + m\, \text{sen}^2\theta) - m((M + m)\, g\, \text{sen}\theta \cos\theta - m\, d\theta'^2 \text{sen}\theta \cos^2\theta - F \cos^2\theta) + F(M + m\, \text{sen}^2\theta))/(M + m\text{sen}^2\theta)$$

$$(M + m)\, x'' = Mm\, d\theta'^2 \text{sen}\theta + m^2 d\theta'^2 \text{sen}^3\theta - m(M + m)\, g\, \text{sen}\theta \cos\theta + m^2 d\theta'^2 \text{sen}\theta \cos^2\theta + mF \cos^2\theta + FM + mF\, \text{sen}^2\theta)/(M + m\, \text{sen}^2\theta)$$

$$(M + m)\, x'' = Mm\, d\theta'^2 \text{sen}\theta + m^2 d\theta'^2 \text{sen}\theta - m(M + m)\, g\, \text{sen}\theta \cos\theta + m\, F + FM)/(M + m\text{sen}^2\theta)$$

$$(M + m)\, x'' = (M + m)(m\, d\theta'^2 \text{sen}\theta - m\, g\, \text{sen}\theta \cos\theta + F)/(M + m\, \text{sen}^2\theta)$$

$$x'' = (m\, d\theta'^2 \text{sen}\theta - m\, g\, \text{sen}\theta \cos\theta + F)/(M + m\, \text{sen}^2\theta)$$

O princípio dos trabalhos virtuais, consequência dos deslocamentos virtuais, utilizado no modelamento clássico, na ótica newtoniana, que apenas estabelece as condições de equilíbrio estático de sistemas, não pode ser aplicado em problemas dinâmicos de multicorpos, a não ser que fosse estendido através de outro princípio, o que mais tarde foi realizado por d'Alembert [Bib.5.2]. Esse procedimento nos leva a montar as equações básicas que permitirão obter os dados ligados ao seu comportamento dinâmico e que futuramente permitirão implementar o seu controle, que será tão real quanto a realidade do modelo obtido.

A eficiência da formulação newtoniana é devida à sua natureza primitiva, que não eleva tanto o tempo de processamento, mantendo o custo em O (n^2), e não é tão maior quanto o número de corpos acoplados e seus graus de liberdade.

Domínio do pêndulo

Para manter o pêndulo em cima do carro, pode-se selecionar uma força proporcional a **θ**:

$$F = k\, \theta,$$

E se obtém:

$$d\theta'' = ((M + m)\, g\, \text{sen}\theta - m\, d\theta'^2 \text{sen}\theta \cos\theta - k\, \theta \cos\theta)/(M + m\, \text{sen}^2\theta)$$

$$x'' = (m\, d\theta'^2 \text{sen}\, \theta - m\, g\, \text{sen}\theta \cos\theta + k\, \theta)/(M + m\, \text{sen}^2\theta)$$

NOTA: Uma força proporcional a **senθ**: $F = k\, \text{sen}\theta$, é quase tão boa...

Solução para ângulos pequenos

Para ângulos pequenos, considera-se $\text{sen}\theta = \theta$, $\cos\theta = 1$ e se desprezam as potências de **θ** superiores a 5.2.

$$d\theta'' = ((M + m)\, g\, \theta - k\, \theta - m\, d\theta'^2 \theta)/M$$

$\boldsymbol{\theta}$ é sensivelmente sinusoidal, logo $\theta'_{max} = \omega\,\theta$ e $\theta''_{max} = \omega^2\,\theta_{max}$, logo $\boldsymbol{\theta'^2\,\theta}$ vale, no mínimo, $\omega^2\,\theta_{max}{}^3$, o que é insignificante em comparação com $\boldsymbol{\theta''}$, se $\boldsymbol{\theta}$ for pequeno.

Pode-se desprezar o último termo e se tem que, se $d\theta'' = ((M + m)\,g\,\theta - k\,\theta)/M$, então:

$$\boldsymbol{\theta'' + (k - (M + m)g)/(M\,d)\;\theta = 0}$$
$$\boldsymbol{x'' = (k - m\,g)/M\;\theta}$$

Solução da equação para o pêndulo

$$\boldsymbol{\theta'' + (k - (M + m)g)/(M\,d)\;\theta = 0}$$

Logo:

$$\boldsymbol{\theta = \theta_m\,\cos(\omega\,t) \quad \text{com} \quad \omega = ((k - (M + m)g)/(M\,d))^{1/2}}$$

E de período $T = 2\pi/\omega = 2\pi(M\,d/(k - (M + m)g))^{1/2}$

$$\boldsymbol{T = 2\pi(M\,d/(k - (M + m)g))^{1/2}}$$

T não é positivo a não ser que $k > (M + m)g$

Logo, o pêndulo não pode ser mantido, a não ser que:

$$\boldsymbol{k > (M + m)g}$$

Ou melhor: $m < k/g - M$

Solução da equação para o carro

$$\boldsymbol{x'' = (k - m\,g)/M\;\theta = (k - m\,g)/M\;\theta_m\;\cos(\omega\,t)}$$

Logo:

$$x = -(k - m\,g)/(M\omega^2)\;\theta_m\;\cos(\omega\,t) = -(k - m\,g)/(k - (M + m)g)\;d\;\theta_m\;\cos(\omega\,t)$$
$$\boldsymbol{x = -(k - m\,g)/(k - (M + m)g)\;d\;\theta = -(k - m\,g)/(k - (M + m)g)\;d\;\theta_m\;\cos(\omega\,t)}$$

Modelamento Matemático Newton/Euler/ Bryant?

Uma alternativa para derivar equações mais eficientes de movimento foi o desenvolvimento de algoritmos mais eficientes para o processamento de forças/Torques generalizados baseados nas equações de Newton/Euler para o movimento. Excluindo a dinâmica do atuador e fricções, as equações da dinâmica do sistema resultante são um conjunto de equações recursivas no sentido direto e inverso. A recursão direta propaga informação da cinemática como velocidades lineares e angulares, acelerações lineares e angulares do Centro de Massa, sendo que a recursão inversa propaga as forças e momentos exercidos sobre a estrutura da cadeira em questão do Centro de Gravidade (CG) em relação ao referencial inercial escolhido.

A derivação da formulação de Newton / Euler é simples sob o aspecto computacional, mas um pouco confusa para a interpretação no sentido em que envolve produto de vetores, escondendo a visão de estados, como era em Lagrange.

Devido à simplicidade computacional dessa formulação, seu custo é baixo em O(n). Acredita-se que seja possível a implementação do algoritmo de controle em tempo real a partir do ponto em que o protótipo real esteja disponível para o teste e cujos resultados sejam confrontados com aquele simulado.

Modelo Newton/Euler/Bryant?

É cabível, para este modelo, desenvolver a relação matemática necessária de um ponto g em um sistema de coordenadas que sofre rotação em um referencial inercial fixo e, então, estender o conceito para incluir a translação **d** em relação ao seu referencial inercial. Assim, considerando essa extensão em relação ao referencial fixo:

$$\mathbf{r} = \mathbf{r}^* + \mathbf{d}$$

Onde: **r** e $\mathbf{r}^*$ são os vetores que caracterizam a posição de um ponto **G** determinado do sistema em translação e rotação? Em relação ao referencial inercial. Assim o vetor velocidade:

$$v(t) = \frac{dr}{dt} = \frac{dr^*}{dt} + \frac{dd}{dt} = v^* + v_d$$

Da mesma forma, a aceleração do ponto G se desenvolve da forma:

$$a(t) = \frac{dv(t)}{dt} = \frac{d^2 r^*}{dt^2} + \frac{d^2 d}{dt^2} = a^* + a_d$$

No final, somos conduzidos a equações recursivas generalizadas da forma:

<u>No</u> <u>sentido</u> <u>direto</u>: para i = 1 ... n ;

<u>No</u> <u>sentido</u> <u>inverso</u>: para i = n ... 1;

Onde:

De acordo com esse procedimento, desenvolveu-se o modelo a seguir e se obtiveram as forças e torques para os seis graus de liberdade deste sistema que, propositalmente, como foi dito, é mais generalista, podendo, posteriormente ser simplificado pela imposição de condições iniciais e restrições, como se verá mais adiante.

Modelamento Matemático Lagrangeano

Como sistemas com vínculo não possuem deslocamentos virtuais independentes, Lagrange reescreveu o princípio de D'Alembert em termos de um número mínimo de coordenadas, ditas generalizadas, de forma que, os coeficientes dos deslocamentos virtuais podem ser anulados separadamente, obtendo-se um sistema de equações diferenciais ordinárias, denominadas equações de Lagrange. Uma formulação simples e sistemática.

Sendo as equações de Lagrange escalares, sua derivação é mais simples que a das equações de Newton. Adicionalmente, apenas os deslocamentos e velocidades precisam ser

calculados, excluindo-se a aceleração. Além disso, as forças de reação que não produzem trabalho, não precisam ser incluídas na formulação das equações do movimento, como era feito para as equações de Newton. Para o caso da cadeira de rodas em questão, que possui um número de graus de liberdade elevado, escrever o lagrangeano de um dispositivo dessa natureza é uma tarefa bem árdua. Apenas poderia ser inviabilizada se o número de derivadas simbólicas fosse muito elevada, o que não será o caso, levando-se em conta as restrições que serão impostas, para processamento computacional [Bib.5.3]. Outra forma de diminuir o custo do processamento pela formulação de Lagrange sera ignorar as forças devidas a Coriolis e centrífuga. Contudo, estas podem ser significativas quando o dispositivo, a cadeira, executa os movimentos em velocidades rápidas, principalmente visando ao equilíbrio.

A formulação de Lagrange fornece equações explícitas de estado e pode ser usada tanto para resolver problemas de dinâmica direta, ou seja, dados os torques/forças, as equações nos fornecem as acelerações que são, então, <u>integradas</u> (ver ODE) para retornar as coordenadas generalizadas e suas velocidades, como para a dinâmica inversa que, dadas as coordenadas generalizadas e suas duas primeiras derivadas no tempo (velocidades e acelerações), retornam as forças e torques generalizados.

A pouca eficiência da formulação lagrangeana é devida à sua natureza matricial, que eleva o tempo de processamento, elevando o custo $O(n^4)$, e é tão maior quanto o número de corpos acoplados e seus graus de liberdade.

Modelo Lagrangeano

Partindo-se da formulação da energia cinética (K) e da energia potencial (P), obtém-se a função lagrangiana (L) da forma:

$$L = K - P$$

Onde a derivada da expressão acima determina a equação do movimento pela determinação das forças (F_i ou torques T_i) generalizadas em função das coordenadas generalizadas (q_i) e suas primeiras derivadas $(\dot{q}_i)$, conforme se vê abaixo:

$$\frac{d}{dt}\left[\frac{\partial L}{\partial \dot{q}_i}\right] - \frac{\partial L}{\partial q_i} = F_i \qquad \text{para } i = 1,2,...,n$$

No final, somos conduzidos a uma equação generalizada da forma:

$\boldsymbol{\tau}\mathbf{(t)} = \mathbf{D}(\mathbf{q}(t))\,\mathbf{q''}(t) + \mathbf{h}(\mathbf{q}(t), \mathbf{q'}(t)) + \mathbf{c}(\mathbf{q}(t))$ para i = 1 ... n;

Onde:

τ(t) = Vetor de torques generalizados;
D(**q**) = Matriz simétrica de aceleração inercial de;
q(t) = Vetor de coordenadas generalizadas cuja derivada primeira é;
q'(t) = Vetor de velocidades relacionadas cuja derivada primeira é;
q"(t) = Vetor de acelerações relacionadas;
h(**q**, **q'**) = Vetor de forças não lineares devido à Coriolis e centrífugas.
c(**q**) = Vetor de forças devido à gravidade.

Simulação com o modelo Lagrangeano

Aproximação Lagrangeana

$L = Ec - Ep = 1/2\ m v^2 + 1/2\ Mv_c^2 - mg\ d \cos\theta$ (altura nula no eixo do pêndulo)

Onde: $v^2 = x_x^2 + v_y^2 = (d\theta'\cos\theta + x')^2 + (-d\theta'\text{sen}\theta)^2 = d^2\theta'^2 + x'^2 + 2\ d\ x'\ \theta' \cos\theta$

$L = 1/2\ m\ d^2\theta'^2 + 1/2\ m\ x'^2 + m\ d\ x'\ \theta' \cos\theta + 1/2\ M\ x'^2 - mg\ d \cos\theta$

Fórmulas de Lagrange:

Se: $d(dL/d\theta')/dt - dL/d\theta = 0$

Então: $m\ d^2\theta'' + m\ d\ x'' \cos\theta - m\ d\ x'\ \text{sen}\theta\ \theta' + m\ d\ x'\ \theta'\ \text{sen}\theta - mg\ d\ \text{sen}\theta = 0$

Logo:

$m\ d^2\theta'' + m\ d\ x'' \cos\theta - mg\ d\ \text{sen}\theta = 0$

$\mathbf{d\theta'' = g\ \text{sen}\theta - x''\cos\theta}$ **(Relação 1)**

$d(dL/dx')/dt - dL/dx = F$

$m\ x'' + m\ d\ \theta'' \cos\theta - m\ d\ \theta'^2\ \text{sen}\theta + M\ x'' = F$

Logo:

$\mathbf{(M + m)\ x'' = m\ d\theta'^2\text{sen}\theta - m\ d\theta'' \cos\theta + F}$ **(Relação 2)**

Substitui-se **x''** na relação **1**:

$(M + m)\ d\theta'' = (M + m)\ g\ \text{sen}\theta - m\ d\theta'^2\text{sen}\theta\cos\theta + m\ d\theta'' \cos^2\theta - F\cos\theta$
$(M + m\ \text{sen}^2\theta)\ d\theta'' = (M + m)\ g\ \text{sen}\theta - m\ d\theta'^2\text{sen}\theta\cos\theta - F\cos\theta$
$\mathbf{d\theta'' = ((M + m)\ g\ \text{sen}\theta - m\ d\theta'^2\text{sen}\theta\cos\theta - F\cos\theta)/(M + m\ \text{sen}^2\theta)}$

Substitui-se **dθ''** na relação **2**:

$(M + m)\ x'' = m\ d\theta'^2\text{sen}\theta - m\ ((M + m)\ g\ \text{sen}\theta\cos\theta - m\ d\theta'^2\text{sen}\theta\cos^2\theta - F\cos^2\theta)/(M + m\ \text{sen}^2\theta) + F$
$(M + m)\ x'' = (m\ d\theta'^2\text{sen}\theta\ (M + m\ \text{sen}^2\theta) - m\ ((M + m)\ g\ \text{sen}\theta\cos\theta - m\ d\theta'^2\text{sen}\theta\cos^2\theta - F\cos^2\theta) + F(M + m\ \text{sen}^2\theta))/(M + m\ \text{sen}^2\theta)$
$(M + m)\ x'' = (Mm\ d\theta'^2\text{sen}\theta + m^2 d\theta'^2\text{sen}^3\theta - m(M + m)\ g\ \text{sen}\theta\cos\theta + m^2\ d\theta'^2\text{sen}\theta\cos^2\theta + mF\cos^2\theta + FM + mF\ \text{sen}^2\theta)/(M + m\ \text{sen}^2\theta)$
$(M + m)\ x'' = (Mm\ d\theta'^2\text{sen}\theta + m^2 d\theta'^2\text{sen}\theta - m(M + m)\ g\ \text{sen}\theta\cos\theta + mF + FM)/(M + m\ \text{sen}^2\theta)$
$(M + m)\ x'' = (M + m)(m\ d\theta'^2\text{sen}\theta - m\ g\ \text{sen}\theta\cos\theta + F)/(M + m\ \text{sen}^2\theta)$

Obtém-se:

$$x'' = (m\,d\theta'^2 \text{sen}\theta - m\,g\,\text{sen}\theta\cos\theta + F)/(M + m\,\text{sen}^2\theta)$$

METODOLOGIA E FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, é apresentado um panorama da cadeira com as suas partes constituintes de forma a fazer uma ligação daquilo que é simulado com o dispositivo fixo.

O desenvolvimento prático da cadeira visa um protótipo a ser construído em chassis de alumínio com duas rodas dianteiras livres do tipo “Castor” e duas rodas diferenciais traseiras, acionadas cada uma por um motor DC de 12 Volts, uma placa baseada em um microcontrolador, um netbook cuja placa mãe é compatível PC, mas com interfaces USB apenas e sensores conforme a Figura 5.0 a seguir.

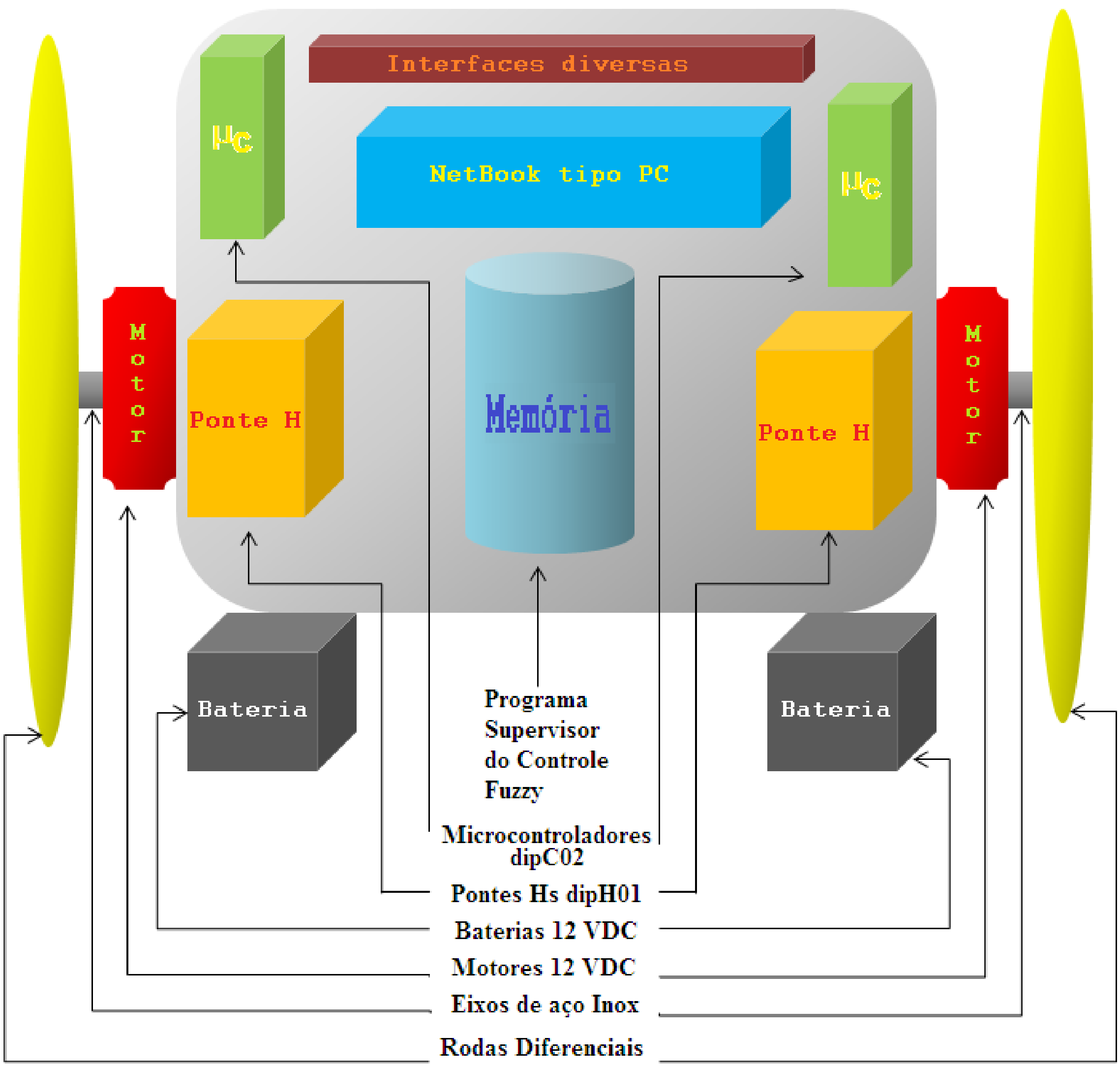


Figura A5.1 – Diagrama em Blocos da cadeira

Sensoriamento é a palavra chave para até o mais simples comportamento de mobilidade. Em um dispositivo que pretende ser robótico, este deve ser capaz de sensoriar e processar este sensoriamento de forma a tomar decisões. Os sensores que estão mais associados com a mobilidade são aqueles que medem a distancia, as mudanças inerciais e os que medem a estrutura externa ao seu redor. Naturalmente, por trás deste sensoriamento, está um algoritmo (programa) que processa convenientemente esta informação.

Sensores

5.1.Introdução.

Um sensor pode ser definido como um dispositivo que muda a sua característica física interna sob a ação de uma grandeza física externa, podendo fornecer direta ou indiretamente, um sinal que pode até mesmo ser elétrico, indicativo daquela grandeza.

Os sensores são os provedores de informação para os processos de automação industrial e robótica.

Em função da informação dada pelos sensores e devidamente adequada pelos transdutores, o controle do processo industrial ou do robô, toma as medidas necessárias para corrigi - lo sob a forma de ações.

Este capítulo faz uma abordagem dos tipos de sensores/transdutores mais utilizados em processos industriais e robótica, assim como adicionalmente, comenta-se sobre o seu princípio físico de funcionamento para que fique mais caracterizado científicamente.

Do ponto de vista da robótica, podemos afirmar que um robô que não possui sensores, é simplesmente uma máquina. Isto é, o robô precisa ter sensores que o transformam de, uma máquina simplesmente, para um dispositivo autônomo que percebe o seu ambiente de trabalho e reage com a lógica, às mudanças ao seu redor.

Os sensores auxiliam um robô na execução de tarefas, ajudando-o a determinar os parâmetros do ambiente que o cerca e dos objetos a serem manipulados, podendo ser classificados como sensores externos do robô e, os que servem para o controle do robô, na determinação de seus parâmetros de movimento e esforços, ou seja, posição, velocidade, aceleração e força, denominados de sensores internos do robô.

Faz-se necessário a implementação de sistemas de controle, onde possam medir a grandeza desejada, transformar esta determinada grandeza em linguagem de máquina, fazer um pós-processamento dos dados coletados, analisá-los, e posteriormente tomar uma decisão. Sensores e algoritmos para interpretar estes parâmetros podem ser altamente complexos. Contudo uma vasta variedade de sensores existe, e estes sensores trazem consigo características de certa forma preocupantes como ruídos, limitações em suas faixas de observação e não poderem serem modelados completamente para efeitos de controle. Para minimizar estes efeitos, recorre-se portanto aos algoritmos para que, através de filtros e adaptações, possamos chegar ao mais próximo possível de um dispositivo ideal.

Entre os parâmetros a serem considerados temos:

Variáveis de medida

Os sinais aqui abordados, são informações que representam variações de uma grandeza física, valores ou até mesmo, estado. Este sinal pode ser classificado como:

Analógico - Quando os valores são contínuos dentro de um intervalo;
Digital - Quando uma codificação binária compõe numéricamente um valor no tempo de amostragem;
“On - Off” - Quando o valor medido assume apenas dois estados.

Características de um sensor
As principais, para um sensor são:

Linearidade - É a relação entre o sinal gerado e a grandeza física de forma que, quanto mais proporcional, mais linear e fiel é o valor; Faixa (ou range) - São os valores limites (inferior e superior) que o sensor pode operar sem causar sua destruição ou imprecisão do valor.

Faixa (ou range) - São os valores limites (inferior e superior) que o sensor pode operar sem causar sua destruição ou imprecisão do valor.

Veja a seguir, um estudo sob a forma gráfica destas características:

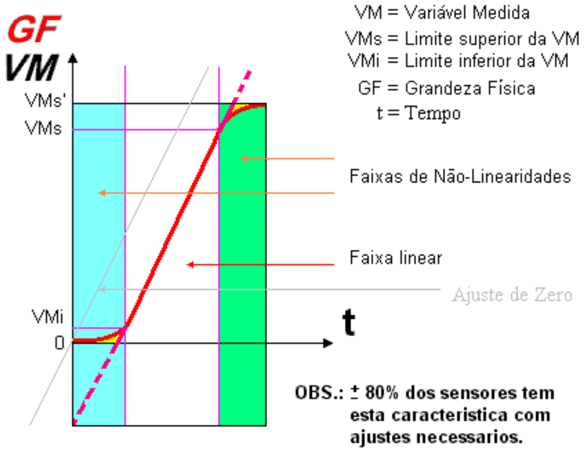
Figura A5.2 - Gráfico das características de um sensor

Obs.: Nas faixas não - lineares, para corrigir são necessárias ações de integração, derivação e deslocamentos.

Geralmente, os sensores são denominados transdutores, pois já convertem a grandeza física medida em uma grandeza elétrica. O transdutor é outro dispositivo que fornece uma resposta de saída elétrica que pode ser medida e processada por um circuito eletro - eletrônico, e que reproduz certas características do sinal de entrada, de mesma natureza ou não. Como exemplo, podemos citar:

A foto - célula que muda sua resistência sob o estímulo da luz;





O foto - transistor que muda o fluxo de corrente sob o estímulo da luz.

São elementos de comando considerados como sensores:
Botão liga - desliga (de contatos NA ou NF) com ou sem trava;
Botão inversor ou comutador;
Chaves Fim - de - Curso (“Reed ou Limit Switch”);

ou então de sinalização como:
Indicadores Luminosos na faixa visível (incandescente, neon (vários tipos), led (vários tipos), etc);

Buzinas, cigarras e sirenes.

São chaves liga - desliga de contatos NA ou NF que possuem um botão de acionamento externo de pressão, usadas em trilhos ou juntas de robôs.

Sensores Óticos São chaves liga - desliga eletrônicas sem contatos que utilizam emissor / receptor a luz visível, IR ou UV e acionamento interno eletrônico. Usadas em trilhos das linhas industriais ou juntas de robôs.

Ultra - Sônicos São dispositivos que emitem um som na faixa ultra - sônica e recebem o eco após o tempo de vôo (t_v), sendo a distância dada pela velocidade do ultra - som (V_{us}) e $t_v/2$.

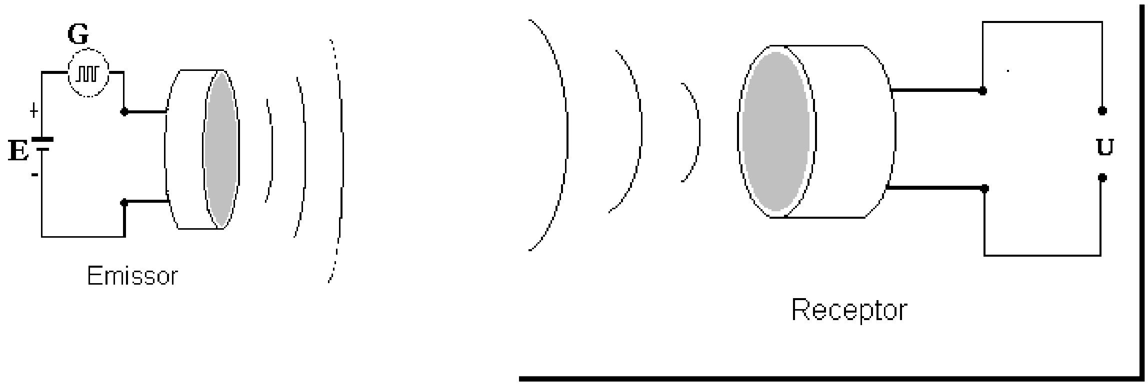


Figura A5.3 – Circuito do Emissor e receptor de Ultra - Som

Estes sistemas são utilizados em máquinas CN, CNC, robôs, linhas de montagem, e etc. O processo de medir uma determinada grandeza através de sensores, e transformá-la em outra grandeza no qual a linguagem de máquina ou computador entenda é denominada popularmente de transdução (após o sensor normalmente vem um transdutor). Uma série de sensores comerciais necessários para este trabalho, será abordado nesta tese.

5.1.1.O sensor de comando vocal

5.1.1.1.Introdução

O principal objetivo deste trabalho foi desenvolver um circuito que permitisse o comando pelo usuário da Cadeira de rodas através da voz.

Esta é a entrada para que o sistema como um todo inicie a ação e os movimentos.

A implementação deste foi possível devido a utilização do Kit Voice Direct ™, cujo funcionamento será descrito mais adiante.

Não foram consideradas neste trabalho, o comando manual, mas algumas modificações a fim de permitir que o comando da cadeira de rodas que era através de um joystick possa ser feito de forma vocal.

Juntamente com o comando por voz foi desenvolvido um outro que possibilitou o controle da cadeira de rodas via microcomputador compatível PC, ou seja, agora a cadeira de rodas poderá ser comandado pelo micro e voz cujos sinais são enviados a cadeira de rodas por transmissão via rádio.

O circuito foi montado dentro de uma caixa de plástica rígido de dimensões 4 X 5 X 7 cm (altura x largura x comprimento).

5.1.1.2.Voice Direct ™

Este kit é basicamente constituído de uma placa contendo uma Rede Neural (ou do inglês ANN – Artificial Neural Network) que permite treinar, aprender e gravar para que posteriormente, possa fazer o reconhecimento das palavras.

A gravação/ reconhecimento são feitos a partir de um microfone de eletreto. A confirmação é enviada ao usuário através de um alto-falante, informando para este se seu comando foi gravado/aceito ou não. A interface da placa com o usuário e composta basicamente de 2 chaves (Treino e Reconhecimento) um alto-falante e um microfone, como dito anteriormente.

Para realizar a gravação de palavras na memória do circuito é utilizada uma entrada específica da placa, chamada "TRAIN".O circuito permite a gravação de até quinze comandos diferentes de, no máximo, 3.2 segundos de duração. Para efetuar a gravação o usuário deve apertar a chave que aciona o modo de treinamento, aguardar o sinal do circuito indicando que este está pronto para armazenar o comando, pronunciar a palavra (de preferência palavras maiores, para não correr o risco de o sistema confundir comandos parecidos e também para minimizar o problema de ruídos no ambiente de uso) e aguardar a resposta do circuito indicando o armazenamento do comando na posição de memória desejada.

Para o reconhecimento é utilizada outra entrada específica chamada "RECOGNIZE". O processo de reconhecimento é bastante simples, basta o usuário apertar a chave que aciona o modo de reconhecimento, dizer um dos comandos anteriormente gravados e aguardar a resposta do circuito, que indica a palavra reconhecida de acordo com a posição em que foi gravada na memória. Por exemplo : Digamos que a palavra "Avançar" tenha sido gravada na posição três da memória. Ao pronunciá-la no modo de reconhecimento o circuito retornará, através do alto-falante, a posição da memória em que a mesma foi gravada, ou seja, três. Além disso, o circuito que possui 8 pinos de saída, "estabelece" cada um desses de acordo com a posição da palavra reconhecida na memória, neste caso o circuito levará para +Vcc, durante um segundo, o pino de saída número três. Palavras que ocupam as posições de memória de 1 até 8 ao serem reconhecidas pelo circuito, “estabelecem” somente um pino de saída, entretanto palavras que ocupam as posições de memória de 9 até 15 “estabelecem” dois pinos de saída ao mesmo tempo, por exemplo: ao reconhecer a palavra que está gravada na posição 9 da memória o circuito aciona os pinos de saída 1 e 8, a palavra 10, pinos 2 e 8, a palavra 11, pinos 3 e 8 e assim por diante, deixando sempre o pino 8 em nível alto e alternando os outros de acordo com a localização das palavras na memória.

O procedimento para limpar a memória também é muito simples, basta acionar as duas chaves de comando ao mesmo tempo que serão apagados todos os comandos gravados anteriormente.

5.1.1.3.Funcionamento do Sistema

Utilizando as saídas geradas pelo Voice Direct ™, foi desenvolvido um programa em C++ que, introduzido na memória do netbook possa ser combinado com a lógica nebulosa e seja enviado um código via interface USB para a entrada em um dos microcontroladores (PIC 16F84) que acionam a roda esquerda e direita, que tratam estes sinais e enviam para os respectivos “drivers” da cadeira de rodas, as ordens recebidas via rádio.

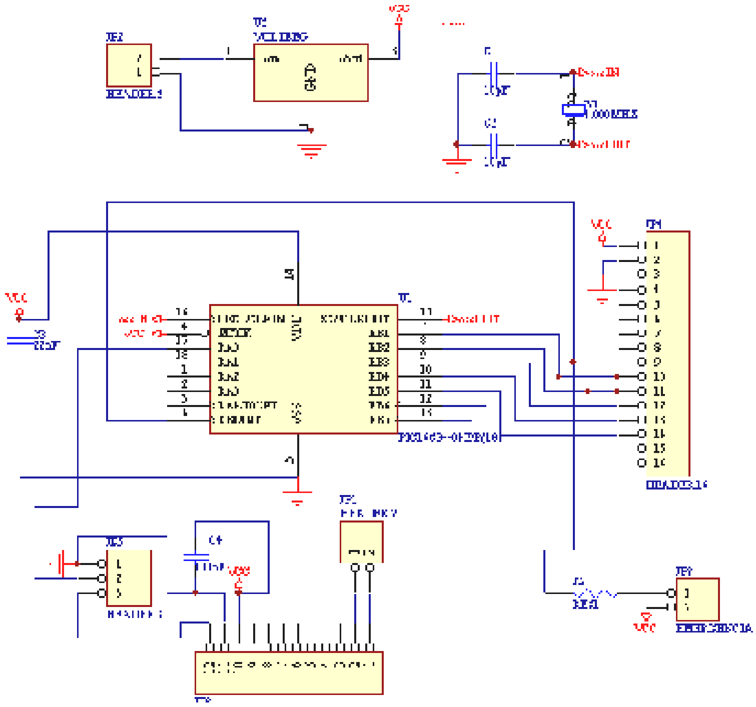


Figura A5.4 – Diagrama elétrico do comando vocal da cadeira

Outro cuidado tomado na confecção da placa foi o de colocar resistores nas saídas do Voicer que estão ligadas em paralelo às saídas de outros circuitos. Esta medida foi tomada para evitar que a corrente gerada ao acionarmos o controle entre na saída do Voicer (já que esta permanece em nível baixo quando não está acionada), o que pode danificar a placa.

Foi implementado também uma chave de emergência no circuito. Inicialmente não existia a intenção de se introduzir uma chave que enviasse diretamente para a cadeira de rodas o comando "PARE", entretanto durante os testes foi constatado que a resposta do circuito a um comando, na maioria das vezes, durava aproximadamente 5 segundos, um tempo muito longo. Este atraso na resposta poderia causar algum acidente, pois se fosse necessário parar a cadeira de rodas rapidamente teríamos que esperar alguns segundos, podendo assim causar choque ou queda, o que poderia danificar a cadeira de rodas e ferir seu usuário.

Para enviar os sinais de comando para a cadeira de rodas foi utilizado um rádio transmissor da marca RADIOMETRIX, o qual tem um alcance em campo aberto de até 50 m.

O sistema de recepção do sinal é composto basicamente de um rádio receptor e entra na interface USB do netbook. Este netbook é quem trata os sinais recebidos e os transmite para os demais controladores PIC da cadeira de rodas.

Conclusão do Projeto

O projeto foi desenvolvido dentro do prazo previsto. O maior obstáculo foi a montagem do circuito dentro da caixa, pois devido ás dimensões reduzidas, foi necessário sobrepor as placas para sobrar espaço para a bateria e o transmissor. É necessário ressaltar que sendo os sinais enviados via rádio, o usuário tem toda a liberdade de levar consigo o sistema, não necessitando de fios para ligá-lo a cadeira de rodas, devendo somente respeitar a distância máxima de 50 m entre o transmissor e a cadeira de rodas .

Um problema observado no projeto foi a sensibilidade do circuito a ruídos vindos do ambiente. Ao pronunciar alguns comandos em ambientes barulhentos, o sistema retorna que a palavra não foi reconhecida e a cadeira de rodas continua respondendo ao comando anteriormente recebido, podendo se chocar com algum obstáculo. Este problema foi parcialmente contornado com a implementação de um chave de emergência, que "para" a cadeira de rodas ao ser acionada.

5.1.2.O sensor Encoder

Encoders são sensores óticos compostos por um disco transparente, marcado com faixas opacas e que gira entre um par foto - emissor - detector.

Podem ser classificados como:

Incrementais - Quando os sinais de saída são um trem de pulsos cuja resolução, está associada à espessura das linhas e a quadratura (direção do giro), à posição dos sensores (em 90^0). Existe um ponto inicial.

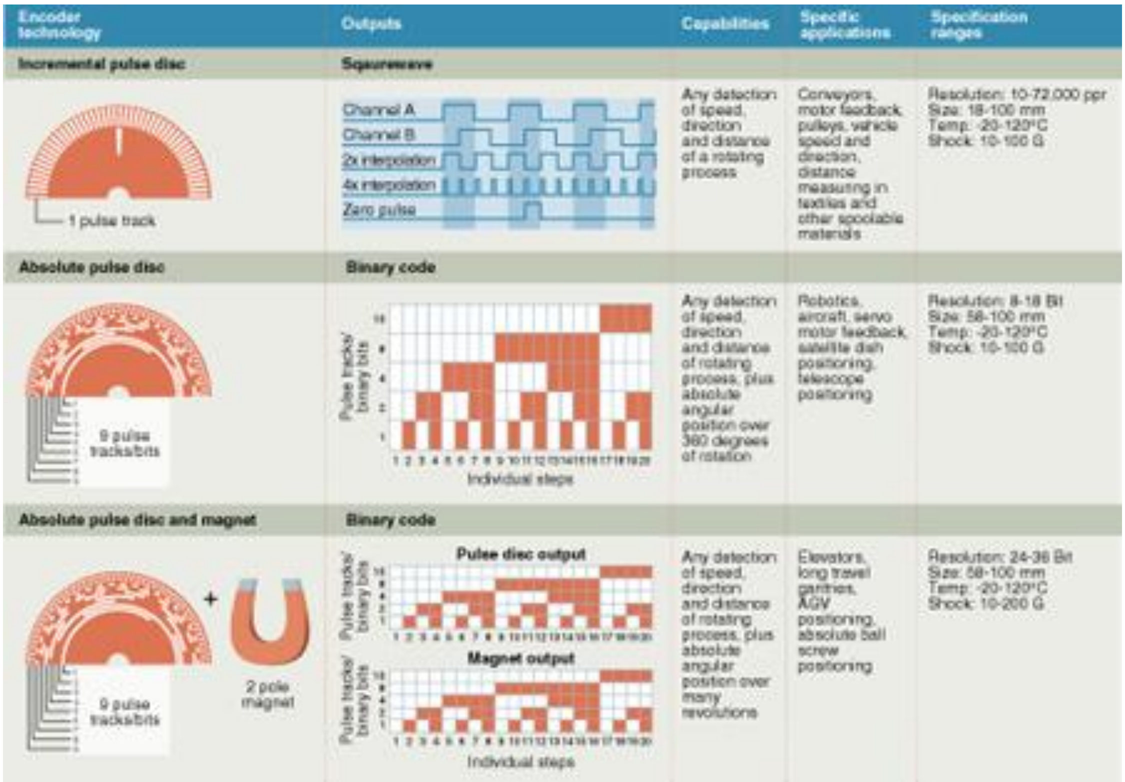
Absolutos - Quando cada posição angular está bem definida de forma única no disco giratório, através do código de Gray, sendo uma vantagem pela não necessidade de determinar um ponto inicial.





FIGURA A5.5 – (a) Vista transparente de um "encoder" e (b) Montagens e encapsulamentos destes.

Tabela A5. 1 – Diagramas de sinais dos diversos "encoders"



Encoder technology	Outputs	Capabilities	Specific applications	Specification ranges
Incremental pulse disc	Squarewave			
1 pulse track	Channel A, Channel B, 2x interpolation, 4x interpolation, Zero pulse	Any detection of speed, direction and distance of a rotating process	Conveyors, motor feedback, pulleys, vehicle speed and direction, distance measuring in textiles and other spoolable materials	Resolution: 10-72,000 ppr Size: 18-100 mm Temp: -20-120°C Shock: 10-100 G
Absolute pulse disc	Binary code			
9 pulse tracks/bits	Pulse tracks/ binary bits; Individual steps	Any detection of speed, direction and distance of rotating process, plus absolute angular position over 360 degrees of rotation	Robotics, aircraft, servo motor feedback, satellite dish positioning, telescope positioning	Resolution: 8-18 Bit Size: 58-100 mm Temp: -20-120°C Shock: 10-100 G
Absolute pulse disc and magnet	Binary code			
9 pulse tracks/bits + 2 pole magnet	Pulse disc output; Magnet output; Pulse tracks/ binary bits; Individual steps	Any detection of speed, direction and distance of rotating process, plus absolute angular position over many revolutions	Elevators, long travel gantries, AGV positioning, absolute ball screw positioning	Resolution: 24-36 Bit Size: 58-100 mm Temp: -20-120°C Shock: 10-200 G

5.1. 2.3.Bourns ™[2]

5.1.3.O sensor Célula de Carga “Strain Gage”

5.1.3.1.Introdução

Existem diversas aplicações para este tipo de dispositivo no qual, é analisada sua empregabilidade para determinar o tipo a escolher. Células de carga são constituídas de um ou mais sensores extensiômetro de resistência elétrica (Strain Gage). Neste relatório experimental realizaram-se medições de massas com o auxílio de uma célula de carga do tipo cilíndrico, no qual foram feitas medições a fim de demonstrar sua funcionalidade e seus componentes bem como fazer uma análise dos resultados.

Neste experimento realizou-se a construção experimental de uma célula de carga do tipo cilíndrico, que utiliza dois sensores extensiométricos, onde foram fabricados seus componentes de acionamento mecânicos e uma parte do seu circuito eletrônico. A fim de demonstrar sua aplicação prática, realizar medidas experimentais e extrair uma análise sobre as mesmas, o objetivo deste trabalho é proporcionar o aprendizado sobre o funcionamento deste sensor e sobre seus componentes. No ensaio realizado em laboratório efetuaram-se cinco séries de medições das massas na célula de carga, a fim de verificar a tensão de saída gerada pelos extensiômetros. Com esta é possível converter em uma grandeza desejada como, por exemplo, peso (N), massa (Kg), pressão (Pa), e etc. A flexão, ou deformação sofrida pela célula é muito pequena, com isso é necessário amplificar o sinal que sai do extensiômetro para ser medido.

5.1.3.2. FUNDAMENTOS TEÓRICOS

5.1.3.2.1. Célula de Carga

Célula de carga é um dispositivo eletromecânico que mede a deformação ou flexão de um corpo e a transforma em uma saída de tensão. O sinal em microvolts é alterado proporcionalmente à medida que aplicamos uma carga em sua estrutura física.

A célula é constituída de um ou mais extensiômetros, e um circuito denominado ponte de Wheatstone. O tipo de aplicação da célula é o fator determinante para a escolha da quantidade de extensiômetros e configuração do circuito da ponte. As figuras 1, 2 e 3 mostram exemplos de aplicações de células.

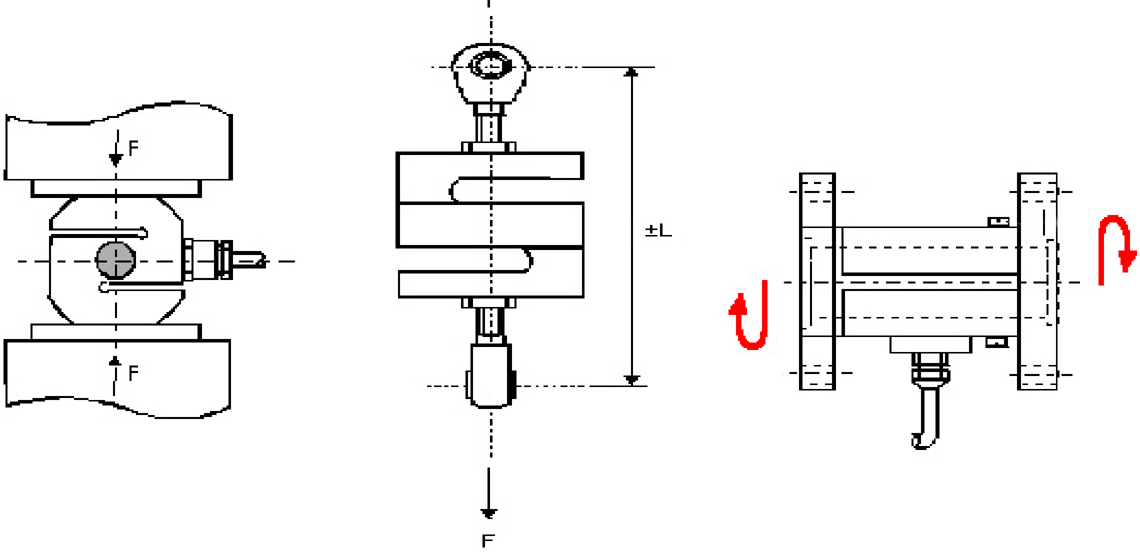
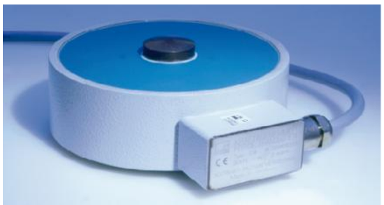


FIGURA A5.6 – Células de (a) compressão (b) torque (c) torque estático.

Neste projeto houve a construção de uma célula de carga em alumínio (Al) do tipo cilindrica, com dois extensiômetros e uma meia ponte de Wheatstone, que tem a finalidade de medir força, peso, ou pressão, como mostram a figura A5.7.





(b)

Figura A5.7 – Sensor extensiométrico(a) Projetado no Laboratório e (b) Comercial. Fonte: HBM® do Brasil

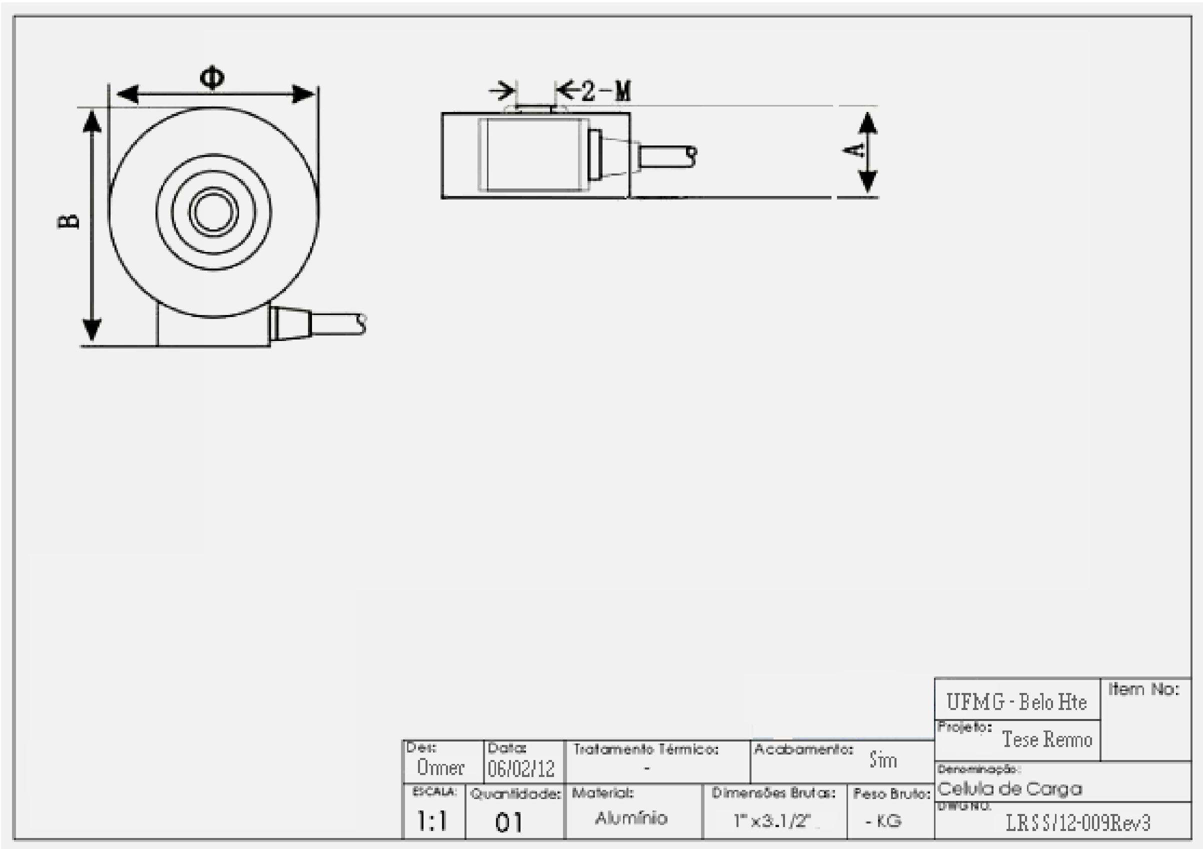


Figura A5.8 – Desenho técnico de construção da célula de carga.

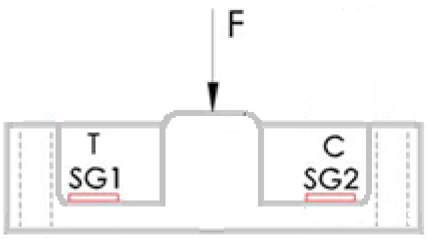


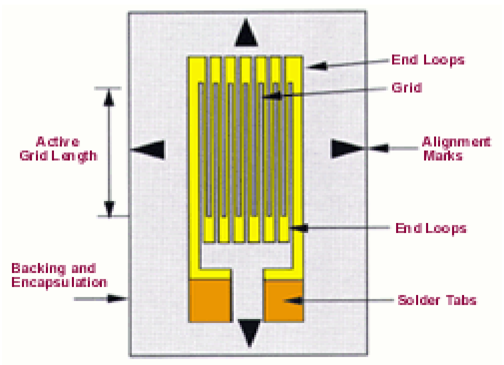
Figura A5. 9 – Esquema de funcionamento do strain gauge no desbalanceamento da ponte.

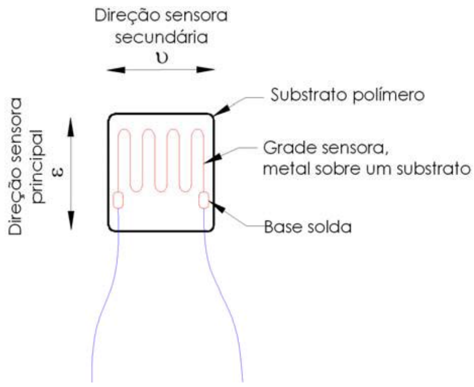
Pode-se observar na figuras A5.9, que esta célula é auto compensada, pois quando aplicada uma força o extensiômetro 1 mede, e o 2 não se altera.

5.1.3.2.2. Extensiômetro ou “Strain Gage”

O extensiômetro de resistência elétrica (Strain Gage), é um resistor elétrico composto de uma grade metálica sobre uma camada isolante de substrato de polímero. Este é colado sobre uma estrutura de teste no qual é sensível a variação de sua resistência em função de uma carga aplicada, podendo – se então estudá-la, medindo e verificando o comportamento de sua estrutura. Estas estruturas por sua vez, apresentam deformações que podem ser monitoradas de diversas formas, dentre as quais: por relógio comparador, por detector eletrônico de deslocamento, por foto - elasticidade, por camada frágil e por “Strain Gage”, dentre outros.

O “Strain Gage” é comumente utilizado pela sua versatilidade. Um sensor de força ou de pressão, por exemplo, nada mais é do que uma estrutura mecânica planejada a deformar-se dentro de certos limites. O extensiômetro realiza a medição em duas direções. A direção principal é a melhor escolha a ser feita, pois possui a maior sensibilidade, ao contrário da direção secundária que é dada pelo coeficiente de poisson (υ), como pode ser observado na figura





5.08. Utilizaram-se dois extensiômetros de 120Ω nesta célula de carga.

Figura A5.10 – Strain Gage.

O fator K fornecido pelo fabricante é dado pela equação 1.

(1)

Onde:

K = fator do extensiômetro
R = resistência inicial
ΔR = variação resistência
L = comprimento inicial
ΔL = variação do comprimento

5.1.3.2.3. Ponte de Wheatstone

O circuito da ponte de Wheatstone é utilizado para medir o desbalanceamento entre os extensiômetros e resistores, causado pela deformação sofrida da estrutura. O desbalanceamento é medido pela variação de tensão e posteriormente transformado na grandeza desejada.

Existem diversos tipos de configurações de ponte, nos quais podemos citar ¼, e ponte inteira, em que a última obtém um melhor resultado, pois possui elevada sensibilidade.

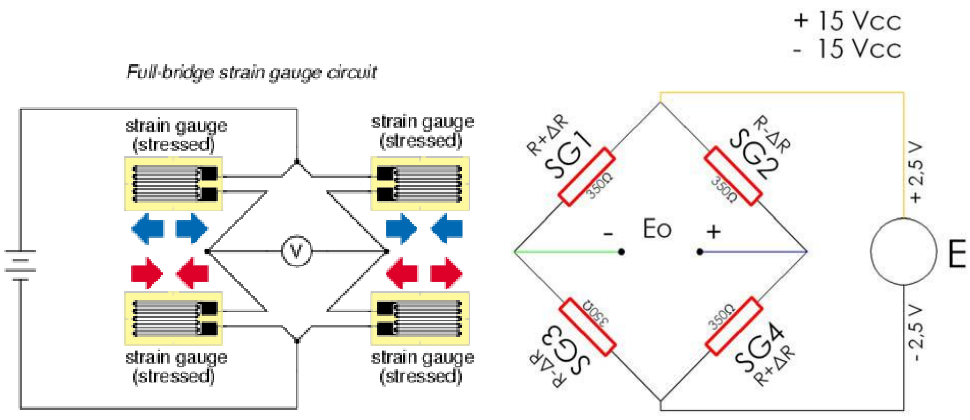


Figura A5.11 – Ponte inteira de Wheatstone utilizada na célula de carga.

A saída Eo da ponte inteira é dada por:

Onde:
ε = Δl/l = deformação relativa = μm/m = microstrain
K = fator do extensiômetro
R = 120Ω

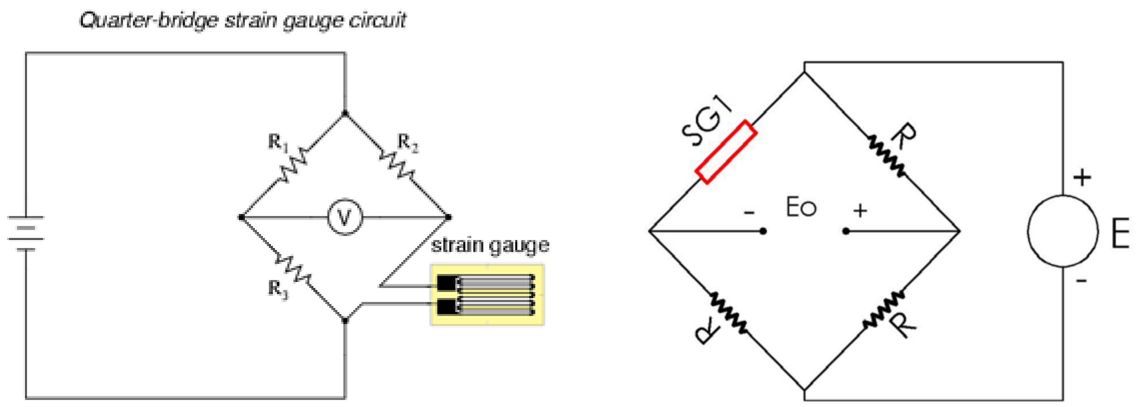


Figura A5.12 – Exemplo de ¼ ponte Wheatstone.

Neste experimento utilizou-se uma ponte ½, como exemplifica a figura 5.13.

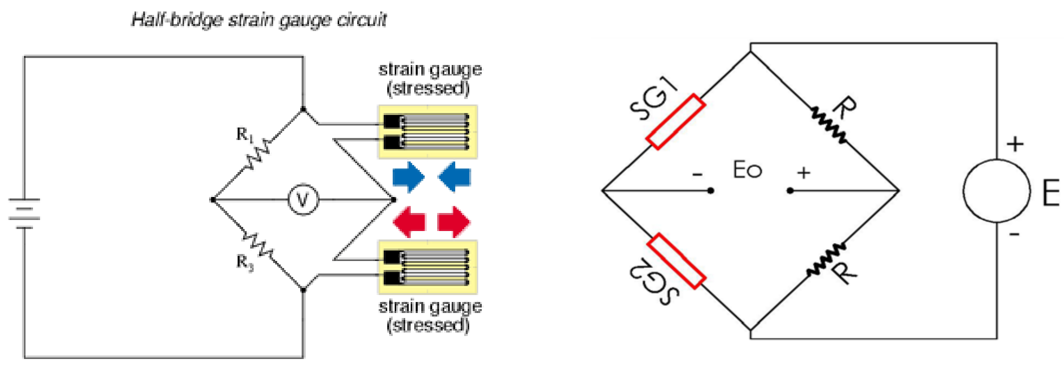


Figura A5.13 – Exemplo de ½ ponte Wheatstone.

5.1.3.2.4. Amplificador para ponte de Wheatstone

O sinal de saída do extensiômetro possui uma variação de tensão muito pequena. Com isso é necessário amplificá-la, para posteriormente efetuar a medição. Utilizou-se um amplificador AO741, do laboratório para efetuar as medições. O amplificador utilizado embora popular e ruidoso, foi projetado para dois extensiômetros de 120Ω com ½ de ponte de Wheatstone, e utilizado neste projeto.

Devido a possibilidade desta situação não ser adequada, pode-se perder sensibilidade na saída do amplificador. O esquema eletrônico do amplificador pode ser observado na figura 5.14, bem como a alimentação da ponte de Wheatstone na figura 5.15.

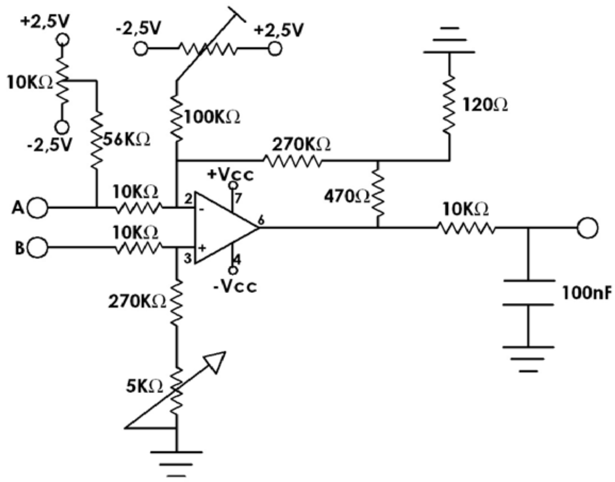


Figura A5.14 – Esquema eletrônico do amplificador OP741.

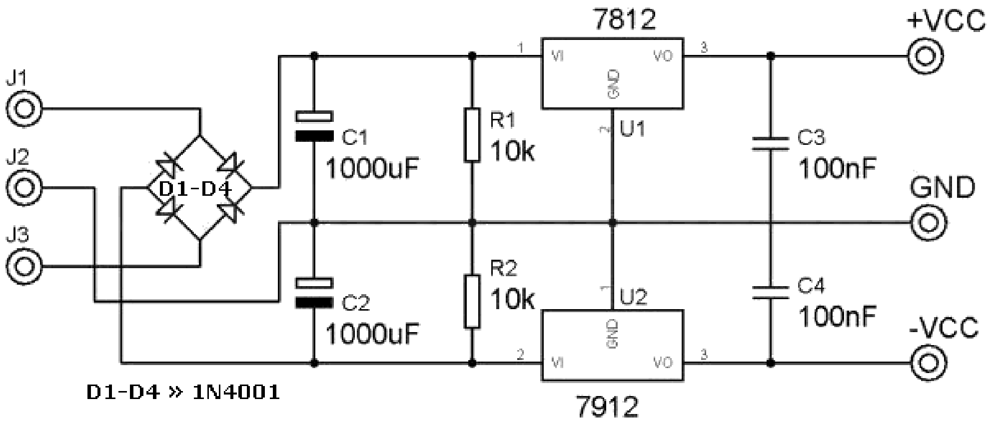


Figura A5.15 – Esquema eletrônico da ligação da fonte ± 12,0V.

5.1.3.3. TESTE

O objetivo do teste é realizar a medição da variação da tensão em função da deformação sofrida pela célula de carga, através de pesos padrão, verificando assim se as suas características atendem ao projeto.

5.1.3.3. 1. Material utilizado

- 01 osciloscópio;

- 01 amplificador OP741 para meia ponte;
- 02 fontes simétricas $\pm$12Vcc;
- 01 jogo de pesos padrão;
- 01 célula de carga;
- 01 multiteste;
- 02 grampos “C”;
- 01 chapa de aço.

5.1.3.3.2. Procedimento e montagem

A célula de carga é fixada em uma chapa de aço para lhe garantir uma melhor fixação e estabilidade na realização das medições. A chapa, por sua vez, é fixada na bancada através de grampos “C”. Foi conectada a célula de carga no amplificador e utilizada duas fontes simétricas para alimentar o sistema. A saída do amplificador ligada ao osciloscópio realizando a calibração do mesmo, e efetuadas cinco séries de medições. Detalhamento da montagem experimental pode ser observado na figura 5.16.



Figura A5.16 – Montagem experimental.

5. 3. Resultados

Tabela A5.2 – Massas x Medidas.

Medida(Volt) Massa (Kg)	1	2	3	4	5	Média
2,00	0,046	0,046	0,046	0,046	0,048	0,046
3,50	0,080	0,080	0,080	0,082	0,082	0,081
4,00	0,095	0,090	0,095	0,095	0,095	0,094
5,50	0,125	0,130	0,125	0,125	0,130	0,127
6,50	0,150	0,150	0,150	0,150	0,150	0,150
8,00	0,190	0,185	0,185	0,185	0,187	0,186
9,00	0,200	0,210	0,210	0,210	0,210	0,208
10,00	0,230	0,230	0,230	0,230	0,230	0,230
11,50	0,260	0,265	0,270	0,270	0,270	0,267
12,00	0,270	0,275	0,275	0,275	0,280	0,275
14,00	0,320	0,330	0,320	0,320	0,330	0,324
15,00	0,350	0,350	0,350	0,350	0,350	0,350
17,00	0,370	0,400	0,400	0,390	0,390	0,390
18,50	0,400	0,420	0,440	0,440	0,440	0,428
21,00	0,440	0,440	0,480	0,480	0,480	0,464

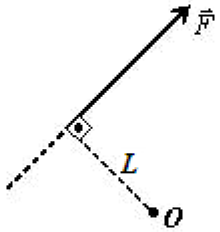
Com os dados coletados foram gerados os gráficos e as tabelas. No gráfico 5.01 houve a realização de uma média das cinco medições, onde é observada uma relação de sua curva praticamente linear.

Realizou-se a medição do ruído na saída do amplificador em torno de 30mV. Uma grande parcela deste ruído pode ser atribuída à inexistência de conexões com bornes adequados para realizar a ligação entre a célula de carga e o amplificador.

5.1.4.Conclusão do Projeto

As células de carga (compostas cada uma de dois “Strains Gages” em ponte de Wheatstone) são os sensores responsáveis pelas indicações das reações. Foram usados um total de quatro células de carga denominados de S_1 a S_4 de acordo com a Fig.8 adiante.

De fato, uma esfera e mola simulam o ser humano e sua movimentação aleatória na cadeira e produzem variações das forças aplicadas e seus respectivos momentos por causa de seus comportamentos físicos da forma:



$$M = \pm F.L \qquad (5.4.7)$$

F – força (em Newton - N)
L – distância (em metros - m)
M – momento da força – N.m

Figura A5.17 - Momento (M) de uma força em relação ao ponto ***O***.

Observação: 1) *F* e *L* são perpendiculares.
2) o momento da força: + sentido anti-horário
- sentido horário

O momento de uma força em relação a um determinado ponto mede a eficiência em causar rotação de um corpo extenso em torno deste ponto. Adicionalmente podemos definir:

Momento de uma Força

O momento de uma força, em relação a um ponto fixo **0**, origem de um referencial inercial, é uma grandeza física, de caráter vetorial, definida pelo produto vetorial do vetor posição do ponto de aplicação da força, em relação ao ponto **0**, pela força aplicada.

Características do momento de uma força

Direção: perpendicular ao plano definido pelos vetores posição e força.
Sentido: dado pelas regras da mão direita ou do saca–rolhas.
Ponto de aplicação: **0**
Norma:

$$\left\|\vec{M}_0(\vec{F})\right\| = \|\vec{r}\|\left\|\vec{F}\right\| \operatorname{sen} \theta \qquad (5.4.8)$$

O ângulo indicado é o ângulo que os vetores posição e força formam entre si. A unidade SI desta grandeza é o N.m.

O momento de uma força tem, na rotação, um papel equivalente ao da força, na translação. O efeito de rotação produzido pela força aplicada a um corpo é medido pelo momento dessa força.

O momento de um sistema de forças, em relação a um ponto fixo **0**. Num referencial inercial, é igual à soma dos momentos das diferentes forças relativamente ao mesmo ponto.

$$\vec{M}_{sist} = \sum_{i=1}^{n} \vec{M}_i \qquad (5.4.9)$$

Momento de uma força e de um sistema de forças em relação a um eixo

O momento de uma força, em relação a um eixo fixo, num referencial inercial, é o vetor deslizante que se obtém projetando no eixo de rotação o momento da força, em relação a qualquer ponto do eixo.

O momento de uma força, relativamente a um eixo, é nulo quando a força e o eixo forem coplanares.

O momento de um sistema de forças, em relação ao eixo em torno do qual gira um corpo, é igual à soma dos momentos das forças exteriores que atuam sobre o corpo, relativamente ao eixo referido.

$$\vec{M}_{zsist} = \sum_{i=1}^{n} \vec{M}_{zi} \qquad (5.4.10)$$

Momento de um binário

Um binário de forças é um sistema de duas forças, simétricas e com linhas de ação paralelas, que produzem efeito rotativo.

Características do momento de um binário

Direção: perpendicular ao plano do binário
Sentido: dado pelas regras da mão direita ou do saca – rolhas
Norma:

$$\left\|\vec{M}_{binário}\right\| = \|\vec{r}_1\|\left\|\vec{F}_1\right\| \operatorname{sen} \theta + \|\vec{r}_2\|\left\|\vec{F}_2\right\| \operatorname{sen} \theta = 2rF \operatorname{sen} \theta \qquad (5.4.11)$$

Notar que as normas dos vetores posição dos pontos de aplicação das forças, relativamente ao ponto fixo **0**, são iguais e que as normas das forças aplicadas também o são.

Um torque devido a forças não altera o movimento de translação do corpo em que atua e produz efeito de rotação sobre o corpo em que é aplicado.

Para o cálculo das coordenadas do CG humano, foi utilizado o equilíbrio das forças fs_1, fs_2, fs_3, fs_4 no plano XY considerando os sensores S_1, S_2, S_3, S_4 aos pares com indicado na Figura 5.17 a seguir:

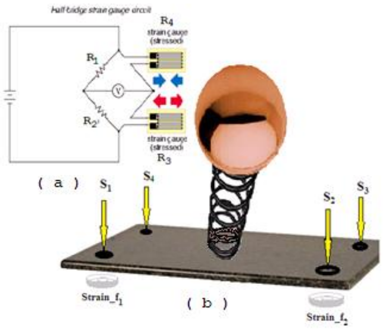


FIGURA A5.18 - (a)Células de Carga com sensores “Strain Gages” e (b) Sistema Esfera-Mola.

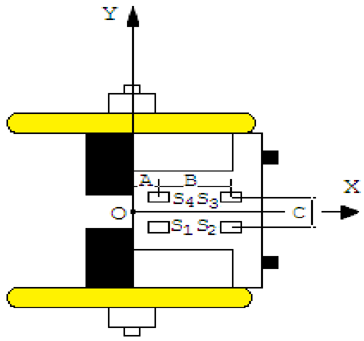


FIGURA A5.19 - Vista do plano XY com os sensores.

Onde:

fs_1, fs_2, fs_3, fs_4 serão as forças nos sensores;
A, B, C, são as distâncias tendo O como referência;

Com este teste mostrou-se que é possível observar as características de sensores extensiométricos em células de carga e que possuem uma larga aplicação entre eles o de sensoriar os movimentos de um usuário de cadeira de rodas, e variam conforme a necessidade. Exibiram-se os componentes necessários para o funcionamento desta célula de carga. Houve a comprovação da variação proporcional de tensão em função da deformação da célula e, observado o comportamento praticamente linear de sua curva. Em função da adequada fixação da célula de carga na bancada de teste através da adição de uma chapa de aço parafusada na célula, obtiveram-se valores satisfatórios nas medições.

Observa-se nos resultados uma pequena variação de tensão em função das massas utilizadas, e pode-se concluir que neste experimento é possível obter precisão melhor do que 1Kg. Limitações podem ser atribuídas ao amplificador utilizado.

É importante ressaltar neste teste, que o laboratório utilizado não possuía as condições adequadas para realizar as medições, pois o amplificador para meia ponte, e extensiômetros de 120Ω utilizado não era adequado aos extensiômetros e ponte utilizados, podendo causar um desbalanceamento inadequado nos resistores, consequentemente desvio na saída de medição, ou até mesmo, perda de sensibilidade.

5.2. Controladores

5.2.0. Considerações

Placas controladoras são partes do hardware de computadores ou circuitos microprocessados/microcontrolados que comandam outras partes internas ou externas da máquina.

Um microcontrolador é um circuito integrado composto por microprocessador programável, memória e interfaces.

5.2.1. Placa PC[TM3] do Sistema Supervisor

5.2.1.1. Introdução

A placa supervisora é um compatível PC® (NetBook da Dell®) adequado para ser o controlador de todo o sistema da cadeira. Como só possui interfaces USB, uma placa conversora foi necessária para compatibilizar com a placa controladora das rodas.

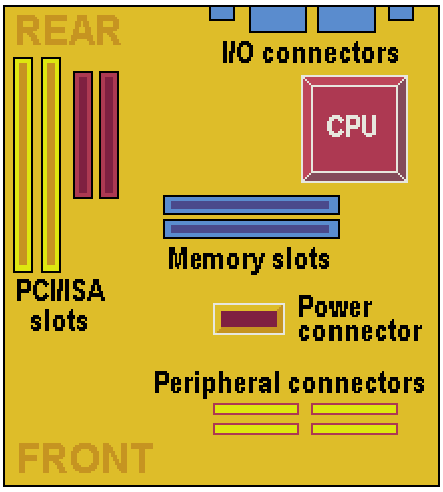


FIGURA A5.20 - Lay-out da placa controladora DELL® baseada no PC®

5.2.1.2. O circuito da placa supervisora da cadeira de rodas

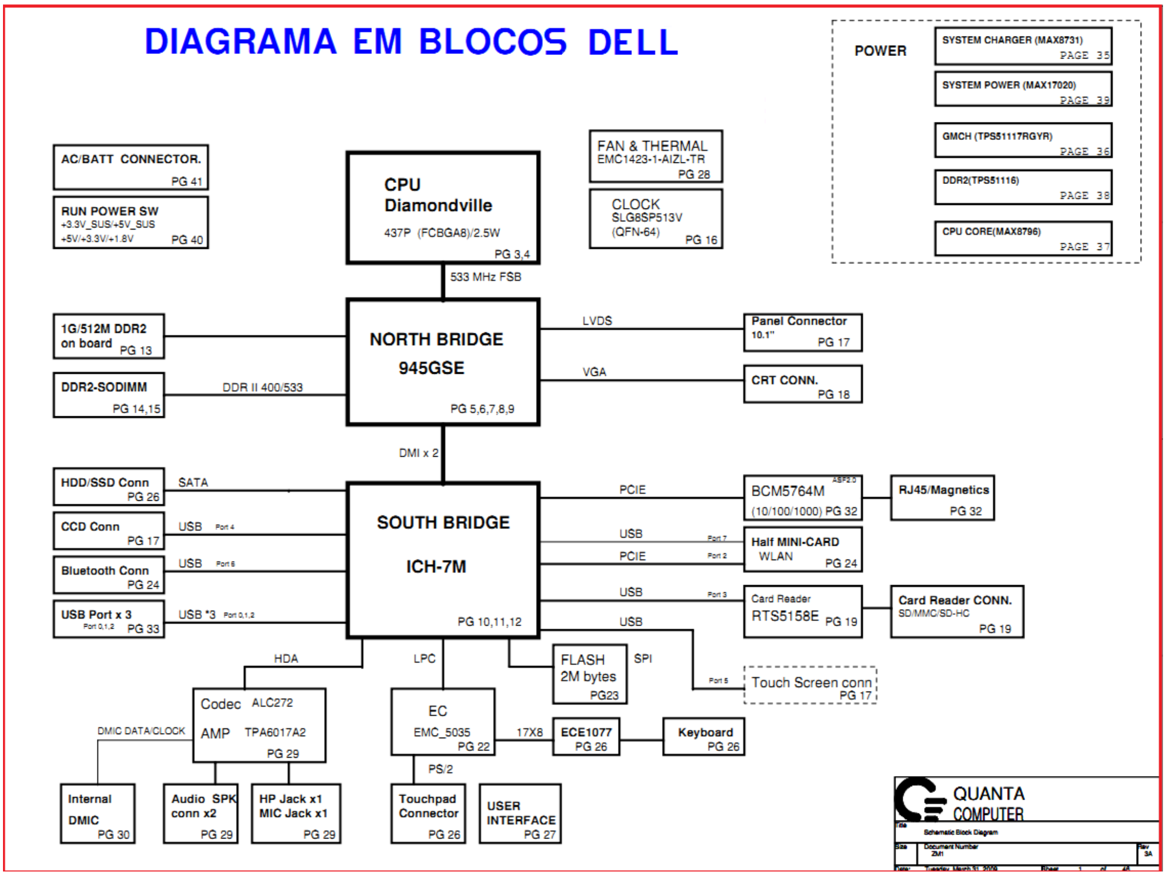
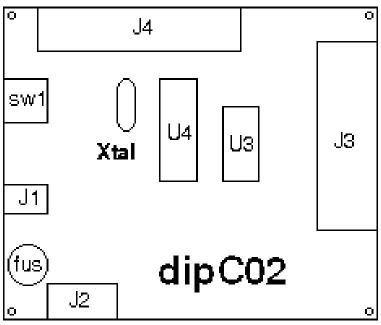


FIGURA A5.21.Diagrama em blocos da placa controladora DELL® baseada no PC®

5.2.2. Placa dipC02^{TM4} do Controlador das rodas da cadeira

5.2.2.1. Introdução

Paralelo aos sinais gerados pelo netbook, podem ser enviados pelos sensores, sinais de comando que entram diretamente na placa controladora do PIC para modificações necessárias nas velocidades e direções através do “driver”. Para isto foi desenvolvido um circuito capaz de enviar tanto os comandos gerados por voz quanto os comandos gerados pelos sensores.



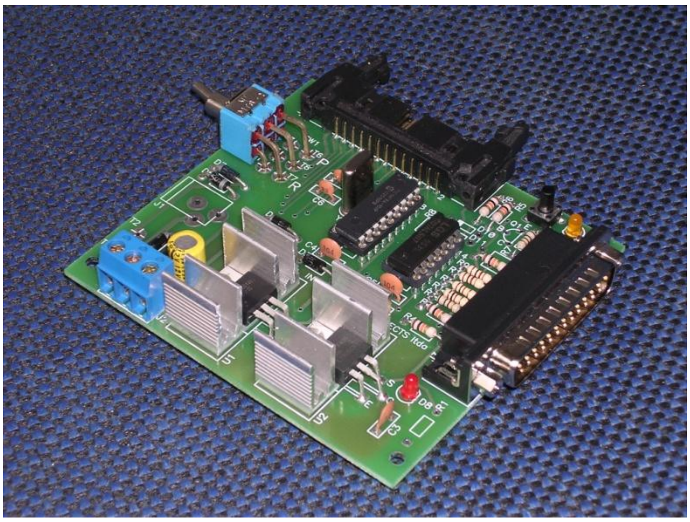


FIGURA A5.22 - Lay-out da placa controladora dos motores fabricada pela Digital International Projects® baseada no PIC16F84®

5.2.2.2. O circuito da placa controladora das rodas diferenciais

A placa controladora dipC02® da Digital International Projects®, é uma unidade eletrônica de automação para trabalhos diversificados menos pesados nas pequenas e médias indústrias e centros comerciais. Faz parte de um sistema modular, composto por diversas placas de interface e drivers para se adequar à uma gama de aplicações que podem estar dentro de áreas como a de lazer, comercial, hospitalar e até em controles de produção industrial com uma certa dose de complexidade.

O microcontrolador utilizado nesta placa é o PIC16F84A® fabricado pela Microchip® (USA) que pode trabalhar em freqüências que variam entre 4 -10 e 20 MHz.

Esta placa é vendida com um disco anexo que contêm manuais, programas, utilitários e um curso sobre este dispositivo, a fim de auxiliar desde o estudante de escolas técnicas aos engenheiros em linhas de produção, nos projetos de seus interesses.

O programa dipAsm, converte o código do programa fonte escrito em linguagem Assembly e desenvolvido no seu computador pessoal para o programa objeto em linguagem binária a ser executado pela placa controladora.

Para que isto seja possível, outro programa, dip16F84®, transmite o código binário do computador pessoal para a placa controladora através de cabo paralelo ou USB que devem ser adquiridos separadamente.

Aconselha-se, que todo o conjunto seja adquirido pois seu funcionamento dependem de todos este conjuntos, podendo o usuário optar por um dos cabos de acordo com as saídas disponíveis em seu microcomputador.

A placa controladora básica da dip pode ser interfaceada tanto pela LPT (J3) como pela USB, tornando mais flexível sua aplicação. O conector na placa é do tipo DB25 e sua conexão com a saída do seu computador pode ser realizada conforme figura a seguir:

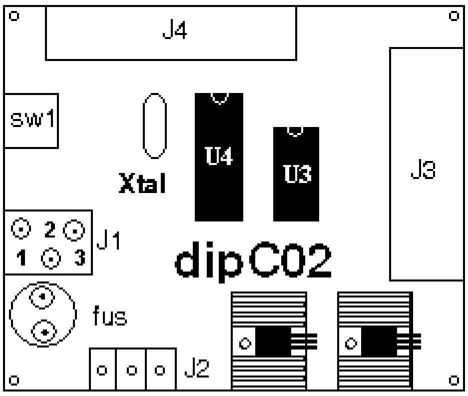


FIGURA A5.23 - Lay-out da placa controladora da Digital International Projects® baseada no PIC16F84®

Temos ainda na placa dipC03® uma interface do tipo RD26 (J4) com níveis compatíveis TTL para conectar esta placa aos processos externos em atuação e/ou aquisição de dados. O esquema é o que se segue:

1.RB0/INT 2.GND 3.RB1 4.GND 5.RB2 6.GND 7.RB3 8.GND 9.RB4 10.+5VCC

11.RB5 12.+5VCC 13.RB6 14.+5VCC 15.RB7 16.+5VCC 17.RA0 18.+14VCC 19.RA1 20.+14VCC 21.RA2 22.+14VCC 23.RA3 24.+14VCC 25.RA4 26.MCLR

A placa dipC02 trabalha com duas alternativas de alimentação. A primeira delas é uma alimentação em corrente alternada de 12 VCA provida por um carregador ligado na rede de alimentação do concessionário em 120/240 VCA. Isto cria uma fixação da placa em seu local de trabalho, não podendo se distanciar da sua fonte de alimentação. A Segunda forma de alimentação em corrente contínua é realizada através de um conector J2 tipo XX para bateria do tipo XX de 9 VDC. Desta forma a placa pode deslocar levando consigo a sua fonte de alimentação. Esta Segunda forma é interessante para dispositivos do tipo robôs ou até mesmo ROVs (submarinos robôs) como aqueles fabricados pela dip.

No projeto, foi necessária a utilização de um regulador de tensão já que o PIC funciona entre 3.6Vdc e 5Vdc e a bateria utilizada era de 9Vdc.

5.3.Motor – A Eletromecânica de Acionamento

5.3.1.Introdução

Motores existem em muitos tamanhos e tipos, mas a função básica de todos eles é a mesma, servem para converter energia elétrica em energia mecânica. Eles podem ser achados em Videocassete, elevadores, aparelhos de CD, brinquedos, robôs, relógios, automóveis, trens do metrô, ventiladores, navios, naves espaciais, condicionadores de ar, refrigeradores, e muitas outras aplicações.

Motores D.C. são motores que giram sob Corrente Direta de uma bateria ou de uma fonte D.C. corrente direta é o termo descrito na eletricidade para uma voltagem constante assim como motores de corrente alternada (A.C.) giram sob Corrente Alternada que oscila com um ciclo fixo entre um valor positivo e negativo. Instalações elétricas convencionais apenas proveem voltagens em corrente alternada.

Quando uma bateria ou fonte D.C. está conectada entre as escovas elétricas de um motor de D.C., o motor converte energia elétrica em trabalho mecânico girando um eixo central.

" O motor elétrico é o mais conveniente de todas as fontes de energia motriz. É limpo e silencioso, parte imediatamente, e pode ser construído grande o bastante para acionar os trens mais rápidos do
mundo ou pequeno bastante para trabalhar em um relógio de pulso".

5.3.2.Características do motor

O motor DC (com escova) é utilizado normalmente para Carga Pesada por ser muito potente para o seu volume. É fácil de usar, pois utiliza dois fios apenas (polaridade $\pm$). Seu torque é proporcional à corrente e o seu estado estacionário é constante.

Segue a Lei da Força de Lorentz de acordo com a equação abaixo:

$F = B \times i \times l$

Onde:

F = Força no enrolamento
B = Campo magnético
i = Corrente
l = No. de Espiras

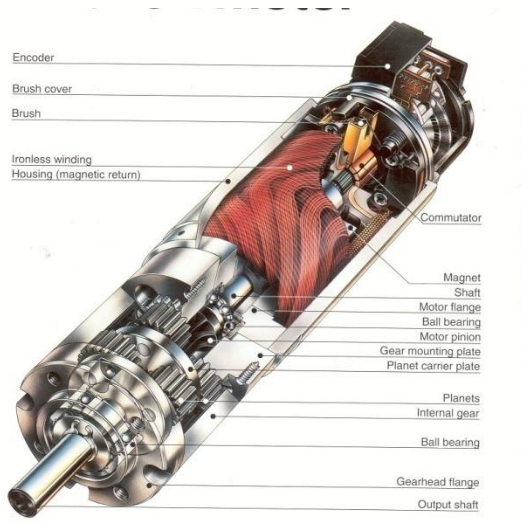


FIGURA A5.24 – Vista interna de um motor onde em (1) .....

E a Regra da mão direita: Dedo indicador em I,
Dedo médio em B,
Polegar na direção de F

A sua velocidade com a carga é proporcional à voltagem, mas para trabalhos mais pesados e de baixa velocidade vai requerer uma redutora conforme a figura abaixo:

No servo controle requer realimentação (ou feedback) de acordo comas suas equações abaixo:

$V = iR + L\,(di/dt) + k_e\omega$

$\tau = k_t\, i$

onde:

V = Voltagem
i = Corrente
R = Resistência
L = Indutância
ω = Velocidade Angular
τ = Torque
k_e = Constante elétrica
k_t = Constante de torque

Força eletromagnética

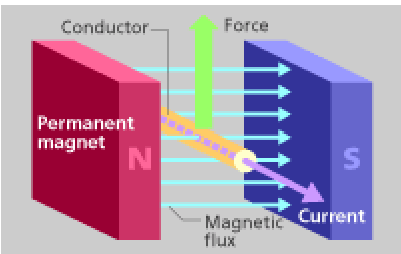


Fig.1

O sentido do fluxo magnético produzido por um ímã permanente é sempre do N-pólo ao S-pólo.
Quando um condutor está colocado em um campo magnético e a corrente flui no condutor, o campo magnético e o atuais interagem-se para produzir a força. A força é chamada “força eletromagnética”.

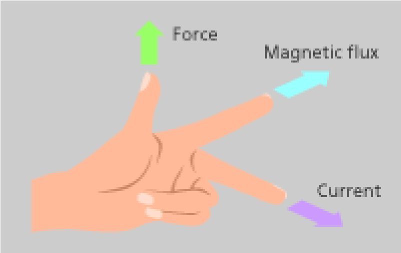


Fig.2

A régua da mão esquerda dos fleming determina o sentido da corrente, da força magnética e do fluxo. Estique o polegar, o dedo do índice e o dedo médio de sua mão esquerda como mostrado em. 2.
Quando o dedo médio é a corrente e o dedo do índice o fluxo magnético, o sentido da força está dado pelo polegar.

Campo do ímã produzido pela corrente

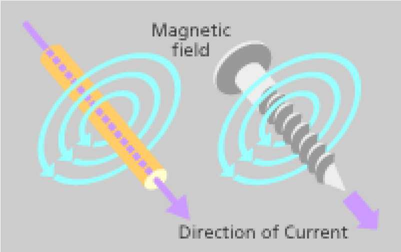


Fig.3

Os campos magnéticos produziram pela corrente e os ímãs permanentes trabalham para produzir a força eletromagnética.
Quando a corrente flui no condutor para o leitor, o campo magnético no sentido do CCW estará produzido em torno do fluxo atual pela régua right-handed do parafuso (Fig.3).

Interferência de uma linha da força magnética

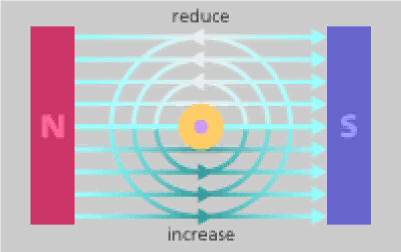


Fig.4

Os campos magnéticos produzidos pelos ímãs atuais e permanentes interferem-se.
A linha da força magnética distribuída no mesmo sentido age para aumentar sua força, quando o fluxo distribuído no sentido oposto agir para reduzir sua força.

Produção eletromagnética da força

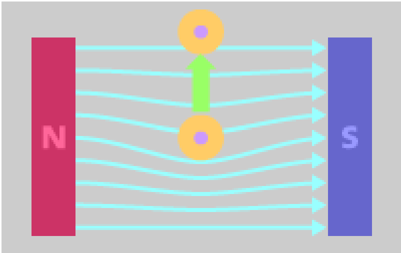


Fig.5

A linha da força magnética tem uma natureza a retornar à linha reta por sua tensão como uma faixa elástica. Assim, o condutor é forçado a mover-se de onde a força magnética é mais forte a onde é mais fraca (Fig.5).

Produção do torque

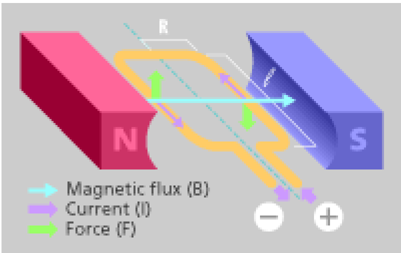


Fig.6

A força eletromagnética é obtida da equação;

$$F\,(\text{force}) = B\,(\text{magnetic flux density}) \cdot I\,(\text{current}) \cdot \ell\,(\text{length of conductor})$$

Fig.6 ilustra o torque obtido quando único-gire o condutor é colocado no magnético arquivado.
O torque produzido pelo único condutor é obtido da equação;

$$T' = F \cdot R$$

- T(torque)
- F (força)
- R (distância do centro ao condutor)

Aqui, há dois condutores atuais;

$$T = 2 \cdot T' = 2 \cdot F \cdot R$$

Também disponível com o motor equipado com o circuito do pulso.

Neste projeto, foram utilizados dois motores com redutoras espelhadas conforme foi dito, para as rodas diferenciais como de um robô móvel conforme a figura abaixo:



FIGURA A5.25 – Conjunto motriz da roda esquerda e direita (a)e o motor de acionamento(b)



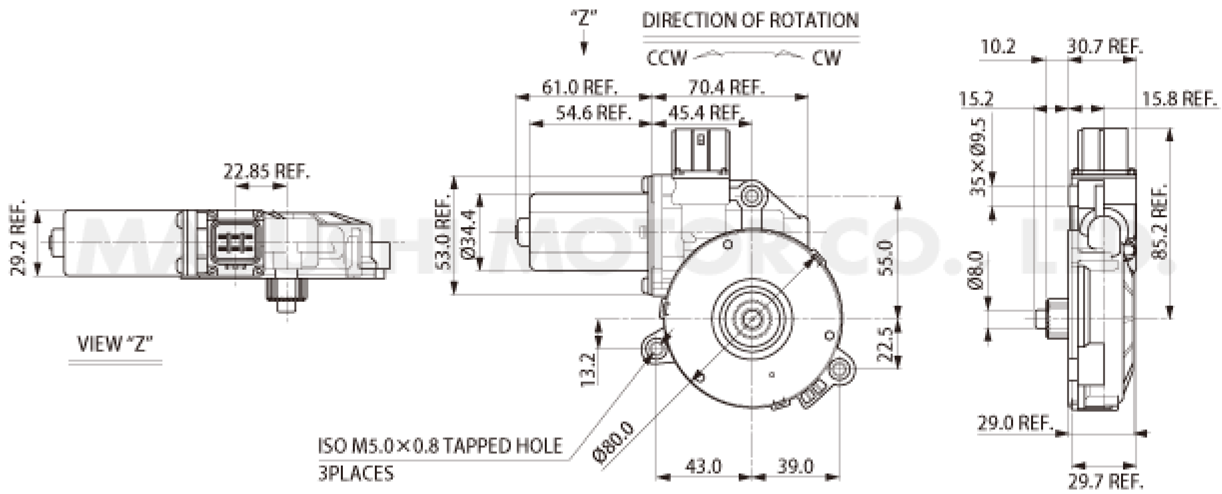
GA-558RN/LN (Powerwindow lift motors)

MABUCHI MOTOR

WEIGHT: 414g (APPROX)

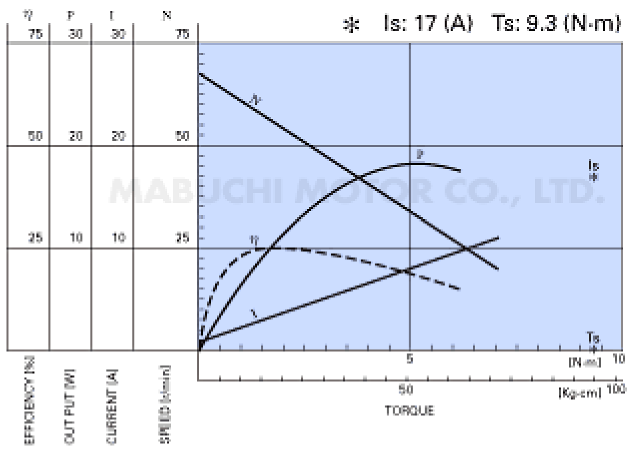
Typical Applications :Automotive Appliances > Power Window Lifter

Also available with the motor equipped with pulse circuit.



UNIT: MILLIMETERS

GA-558RN/LN-4324	12.0V

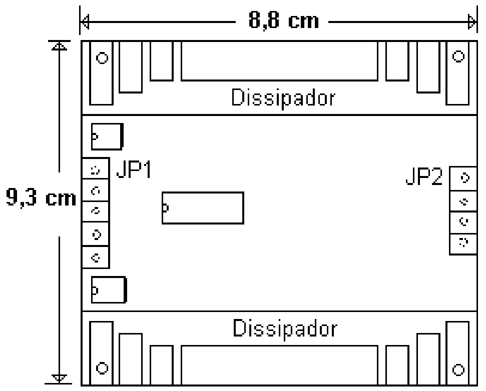


Motores DC requerem drivers para o seu acionamento como uma ponte H, por exemplo.

5.4. Ponte H (“Driver”) - A eletrônica de acionamento

5.4.1.Introdução

Os fabricantes de PWM (Modulação por Largura de Pulso) concorrem para que os seus controladores para motores elétricos sejam os mais eficazes, assim desenvolvemos nosso próprio circuito sem igual experiência técnica, mas que atendeu perfeitamente não só este projeto como outros incorporando aos produtos da *dip* como dipH02 e lembrando que há vários modos de fazer PWM que nós tentamos e rejeitamos (por uma ou outra razão qualquer). Esta seção deverá lhe dar uma boa idéia dos princípios envolvidos e o que fazer, como também o que não fazer! Esta seção é na realidade uma pequena contribuição em controle de motor, visto que este dispositivo foi o mais prático para os acionamentos robóticos na oficina de robótica deste departamento.



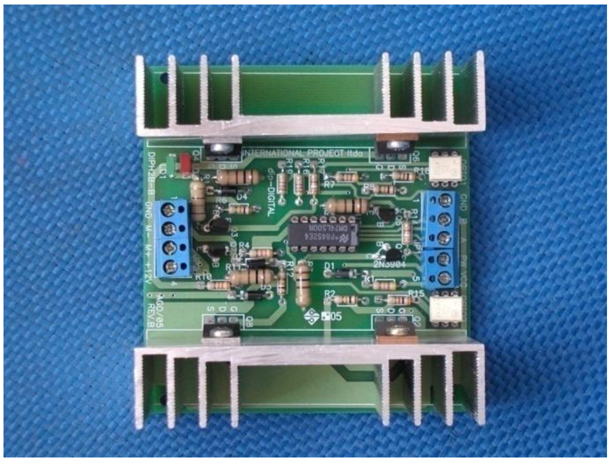


FIGURA A5.26 – Lay-out e a placa do “driver” do motor dipH02® da Digital International Projects®

Um controlador de motor comercial é mais que simplesmente um circuito para alterar a velocidade do motor e a seguir temos um guia de características de controladores que explicam o que a maioria incorpora e por que delas precisam.

Controlar a velocidade de um motor DC utilizado em automóvel (utilizamos o motor DC do vidro elétrico da janela) nós precisamos de uma voltagem ajustável DC que o alimente. Porém, se você utiliza um motor de 12 VDC e o alimenta, o motor começará a acelerar a partir da sua velocidade atual, pois motores não respondem imediatamente e assim levará um tempo pequeno para alcançar a velocidade máxima. Se nós desligarmos a fonte antes da velocidade máxima do motor ter sido alcançada, então este motor começará a reduzir a velocidade, desacelerando. Se nós chavearmos (liga-desliga) de tempo em tempo rapidamente a fonte, o motor correrá de algum modo da velocidade entre zero e velocidade máxima. Isto é exatamente o que um controlador PWM faz, liga o motor em uma série de pulsos. Controlar a velocidade deste motor DC é variar (modular) a largura dos pulsos - conseqüentemente temos a Modulação por Largura de Pulso conforme a Figura 5.26 abaixo:

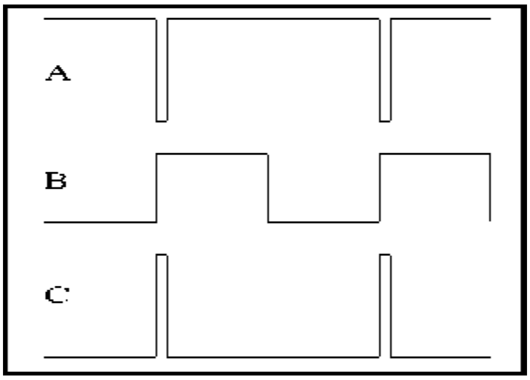


FIGURA A5.27 - Modulação por largura de pulsos

5.4.2. Ponte H para controle PWM de um motor DC

Antes de tentar entender um circuito de ponte H completa, você pode procurar se informar sobre a teoria de controladores de PWM (nos anexos tem algo) que também descrevem como os circuitos de meia – ponte funcionam. Controladores de motor de ponte H também usam PWM! Projetar um circuito de ponte H de PWM que seja seguro e 'à prova de usuário' é bastante difícil, mas existem circuitos confiáveis na praça.

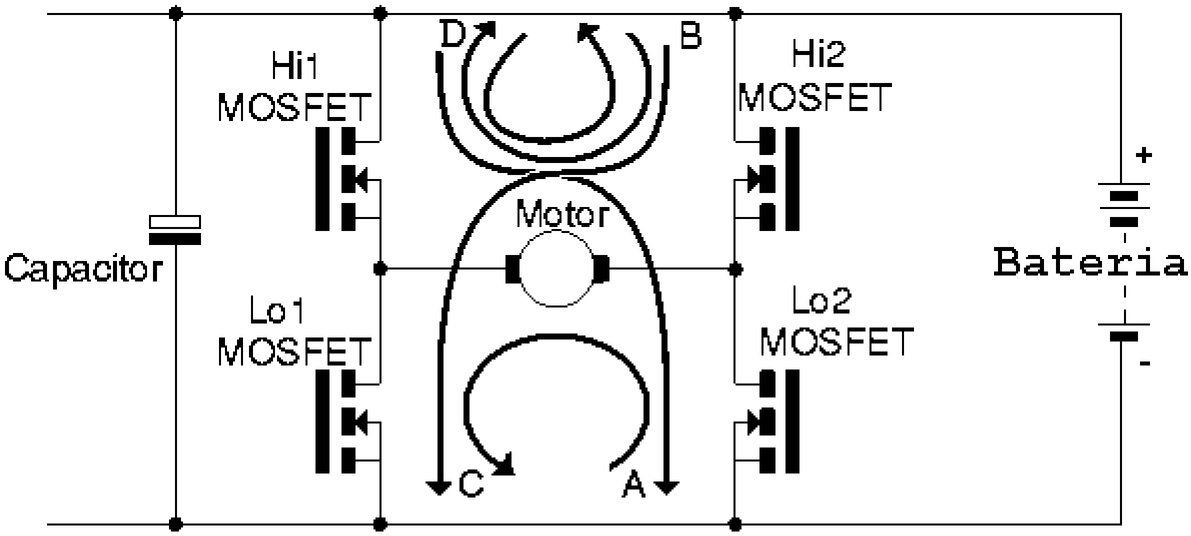


FIGURA A5.28 -Ponte H para controle PWM de um motor DC

O circuito acima mostra uma ponte H completa de quatro MOSFETs. Acionando o motor em sentido adiante, onde a corrente tem que fluir na direção da seta de D para A, saindo do positivo da bateria, passando por Hi1, pelo motor e assim como por Lo2 para o negativo da bateria. Acionando o motor em sentido de contrário, a corrente tem que fluir na direção da seta de B para C. Com o estado atual de ICs e MOSFETs de canal N, somente um ingênuo usaria o MOSFETs do lado-Hi como o elemento ativo de chaveamento principal, assim é normal aplicar PWM aos MOSFETs do lado-Lo, e deixar aqueles do lado-Hi terem um papel mais passivo.

Assim, para sentido adiante, Hi1 estaria permanentemente ativo, Hi2 cortado, Lo1 cortado e Lo2 ativo, talvez selecionado em modo de PWM. Se o PWM é usado, então, quando Lo2 estiver cortado, a corrente do motor ainda precisará continuar fluindo na mesma direção, forçada a fazer assim pela indutância do motor, e usará Hi2 como um diodo de 'flywheel' (

Compensador de flutuação) como mostrado pela seta de D para B. Não importa muito se o MOSFET do lado-Hi que é ativado durante o período de 'flywheel'. Se estiver ativo, então perdas resistivas ocorrem tal como as quedas de um diodo que causariam muito mais perdas.

No sentido inverso é semelhante, com Hi2 permanentemente ativo, Lo1 chaveando e Hi1 compensando (flywheeling). As setas de B para C e de B para D se aplicam.

OBS: Note que se usarmos MOSFET's somente do tipo n precisaremos usar uma fonte elvadora de tensão para as chaves do lado-Hi. Uma fonte elevadora que são capacitores chaveados multiplicadores de tensão provoca conflitos e só funcionaria se a sua saida ao estar chaveando, o motor estiver em velocidade plena e nenhuma chave na ponte estivesse chaveando, caso contrário, falhará.

O resto do projeto depende em 'quão seguro' queremos fazer o circuito. Um controlador comercial em princípio, precisa sobreviver a qualquer coisa que um usuário pode fazer com ele, e requer proteção própria em todos os quatro quadrantes. Assim precisa de algo que limite a corrente em ambas metades da ponte, não só em modo do sentido atuante, mas também em modo de regeneração. Isto acrescenta alguma complexidade!

Assim, o que é freio regenerativo? Dado que utilizaremos um PWM para chavear os transistores de lado-Lo e que a corrente a monitorar é mais fácil de fazer nos MOSFETs do lado-Lo, considere isto como freando regenerativamente adiante. Corrente no sentido dianteira é a seta de D para A. Assim, durante o frear adiante, a corrente é invertida em relação a isto. Em sentido adiante, conforme a figura acima, o lado esquerdo do motor é positivo (foi conectado a bateria + pelo Hi1 ativo). Ao frear adiante, a força contra – eletro - motriz (back emf) do motor está na mesma polaridade como no sentido ativo adiante (como o motor está girando na mesma direção), mas a corrente deve ser invertida (seta de A para C). Assim, o lado esquerdo do motor ainda será positivo e nós temos que ativar permanentemente a chave Lo2, curto - circuitando o lado direito do motor para o negativo da bateria, chaveando Lo1 de tempo em tempo com o sinal do controlador PWM. Deste modo, Lo1 ainda está trabalhando com voltagens positivas (e correntes), tornando ali mais fácil de pensar e mais fácil usar sensores e limitadores de corrente. Se Lo1 está sendo chaveado, então quando está cortado, a energia indutiva do motor ainda forçará uma corrente e será compensada (flywheel) por Hi1 (se lembre que, Lo2 está ativo todo o tempo) para a bateria, dando-se a regeneração (a seta de corrente reversa de D para A).

Para a direção inversa, ao frear que nós precisamos ativar Lo1 em de tempo integral e chavear Lo2.

Há outros esquemas possíveis, mas eles ou envolvem chaveamentos dos lados-Hi utilizando MOSFET's do tipo p ou sensores de correntes negativas. Ou poderíamos pôr um resistor no motor e usar um Amplificador Operacional Diferencial para sensoriar a corrente do motor. Um resistor para sensoriar correntes de talvez vários ampéres não é nem uma artigo comercial comum nem pequeno e é portanto de custo significante! E tem ainda que ser incorporado nitidamente no projeto do controlador.

Pode perceber do que foi mencionado anteriormente, contanto você não adquira a temporização errada, de forma que Hi1 e Lo1 (ou Hi2 e Lo2 no outro sentido) estejam ao mesmo tempo ativos, não importa muito o que você chaveia ou não - a corrente de compensação (flywheel) sempre achará algum lugar seguro para circular.

Claro que o esquema proposto aqui requer que você sinta quando a Força Contra Eletro Motriz (Back emf) do motor está maior que a velocidade exigida, ou seja, quando o motor deveria começar a regenerar, e muda o estado da ponte para fazer isso. Outros esquemas são possíveis, mas eles provavelmente envolvem chaveamento do lado-Hi ativo se você requerer freio regenerativo. Assim este esquema é o método que usamos em nosso

primeiro controlador de ponte completa. Embora seja um projeto simples, ainda é muito popular e muito seguro - sem maiores complicações! Projetos mais simples requerem um pouco mais de análise dos efeitos, combinado com lados-Hi chaveados e uma certa astúcia em prever os limites de correntes regenerativas.

No lay-out do circuito de ponte H completa projetado usamos MOSFET's do tipo p para as chaves do lado-Hi e MOSFET's do tipo n para o lado-Lo neste mesmo esquema e as conexões ficam evidentes abaixo.

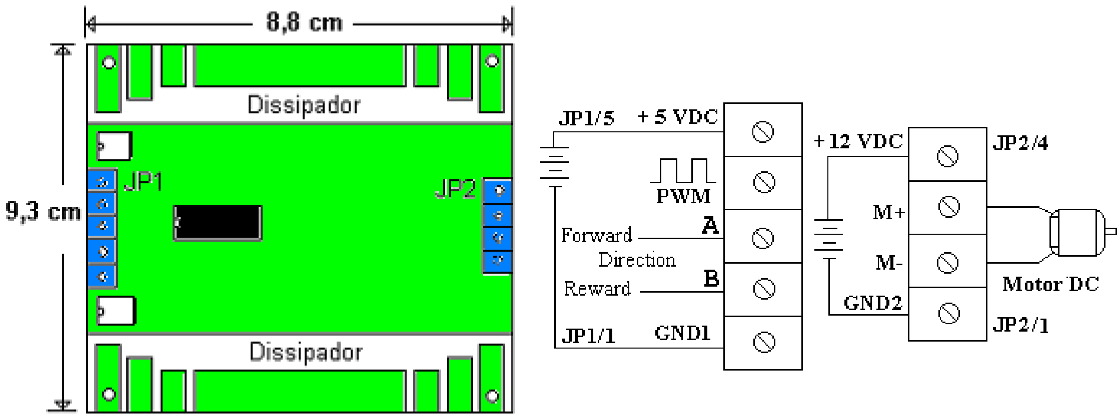


FIGURA A5.29 - Esquema de ligação para Modulação por Largura de Pulsos (do inglês PWM)

5.5.O conjunto estrutural da Cadeira de Rodas

5.5.1.Introdução

Inicialmente como foi dito, a cadeira foi projetada e simulada para atender à determinados comportamentos que elevassem os graus de liberdade alcançados por um deficiente com as cadeiras convencionais. Neste sentido, o desenvolvimento se apoiou em um modelo matemático que contemplasse esses movimentos, no plano e fora dele.

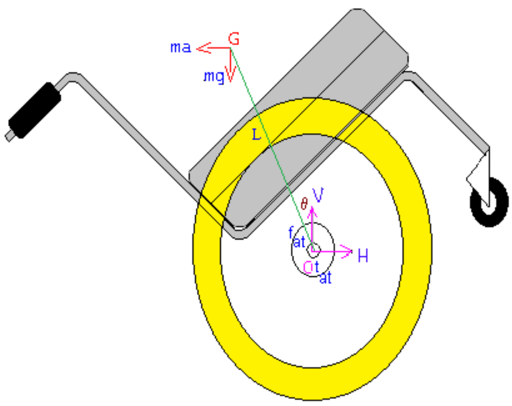


FIGURA A5.30 - Esboço da cadeira de rodas robótica e equilibrista

A cadeira está baseada em uma plataforma de alumínio que possui um movimento de balanço em torno do eixo de duas rodas traseiras pneumáticas que participam do acionamento

de forma independente ou seja, são de tração diferencial, o que permite o conjunto de movimentos estabelecidos tais como: Girar à direita, girar à esquerda, inclinar, avançar e recuar entre outras.

A roda frontal gira livre sem estar vinculada a nenhum acionamento, servindo apenas para o apoio da plataforma no solo sem prejudicar o giro em solo.

O projeto original foi orientado para que a situação do equilíbrio fosse provocada por uma partida com um elevado torque.



FIGURA A5.31 - Montagem da cadeira de rodas robótica e equilibrista

5.5.2.Parâmetros físicos da cadeira

Tab.A5.3. Parâmetros Físicos da Cadeira de rodas Equilibrista

Parâmetro	Descrição	Valor
Mad	Massa do conjunto da roda direita	895 g
Mae	Massa do conjunto da roda esquerda	895 g
Maf	Massa do conjunto da roda frontal	280 g
Mmd	Massa do motor impulsor direito	1565 g
Mme	Massa do motor impulsor esquerdo	1565 g
Mrd	Massa do redutor direito	330 g
Mre	Massa do redutor esquerdo	330 g
Mcc	Massa do chassis da cadeira	1790 g
Mbd	Massa da bateria direita	2580 g
Mbe	Massa da bateria esquerda	2580 g
MCG	Massa do Centro de Gravidade	
l	Largura do chassis da cadeira	0,30 m
c	Comprimento do chassis da cadeira	0,40 m

Cad	Coordenada do conjunto da roda direita	0 x 0 x 210mm
Cae	Coordenada do conjunto da roda esquerda	0 x 0 x -210mm
Caf	Coordenada do conjunto da roda frontal	300 x –20 x 0mm
Cmd	Coordenada do motor impulsor direito	-30 x 117 x 110mm(*)
Cme	Coordenada do motor impulsor esquerdo	-80 x 100 x -110mm(*)
Crd	Coordenada do redutor direito	0 x 0 x -110mm
Cre	Coordenada do redutor esquerdo	0 x 0 x -110mm
Ccc	Coordenada do chassis da cadeira	160 x 0 x 0mm
Cbd	Coordenada da bateria direita	5 x 50 x 32mm
Cbe	Coordenada da bateria esquerda	5 x 50 x -32mm(*)
CCG	Coordenada do Centro de Gravidade	11,3817 x 41,6940 x 0
g	Aceleração da gravidade	9,81 m/s^2
ccs	Coeficiente de atrito seco do eixo do motor DC	0,0005
ccv	Coeficiente de atrito viscoso do eixo do motor DC	0,0005
VARIÁVEIS	Descrição	
xG	Posição linear X do Centro de Gravidade da cadeira	
dxG/dt ou v_x	Velocidade linear X da cadeira	
d^2xG/dt^2 ou dv_x/dt ou a_x	Aceleração linear X da cadeira	
yG	Posição linear Y do Centro de Gravidade da cadeira	
dyG/dt ou v_y	Velocidade linear Y da cadeira	
d^2yG/dt^2 ou dv_y/dt ou a_y	Aceleração linear Y da cadeira	
zG	Posição linear Z do Centro de Gravidade da cadeira	
dzG/dt ou v_z	Velocidade linear Z da cadeira	
d^2z/dt^2 ou dv_z/dt ou a_z	Aceleração linear Z da cadeira	
θ	Posição angular theta da cadeira no plano xy	
$d\theta/dt$ ou w_z	Velocidade angular theta da cadeira no plano xy	
$d^2\theta/dt^2$ ou dw_z/dt ou α_z	*Aceleração angular theta da cadeira no plano xy*	
ϕ	Posição angular phi da cadeira no plano zx	
$d\phi/dt$ ou w_y	Velocidade angular phi da cadeira no plano zx	
$d^2\phi/dt^2$ ou dw_y/dt ou α_y	Aceleração angular phi da cadeira no plano zx	
φ	Posição angular psi da cadeira no plano zy	
$d\varphi/dt$ ou w_x	Velocidade angular psi da cadeira no plano zy	
$d^2\varphi/dt^2$ ou dw_x/dt	Aceleração angular psi da cadeira no plano zy	

ou α_x	

Obs: Todas as coordenadas no sentido XYZ foram tomadas em relação à referência inercial no meio do eixo entre as duas rodas traseiras e diferenciais. Como existe perfeita simetria na estrutura da cadeira, não foi necessário modificar a posição de nenhuma peça para compensar uma diferença das coordenadas devido ao estudo do CG na seção 5.2.

ANEXOS

A01.1-Programa sugerido e abrangente em C para ser adaptado para o controle da cadeira de rodas

CÓDIGO DO PROGRAMA DESENVOLVIDO PELA:

```
/*********************************************************************
**********
Fundação Universidade Federal do Rio Grande
Engenharia de Computação
Projeto de Graduação
Planejamento de Trajetórias para Robôs Móveis
---------------------------------------------------------------------
----------
Guilherme de Lima Ottoni
Orientação: Prof. Dr. Walter Fetter Lages
defs.h - Arquivo de definições de constantes,
declarações de tipos e declaração de variáveis globais da classe
externa
*********************************************************************
*********/
#ifndef _DEFS_H
#define _DEFS_H
#include <pthread.h>
#include <semaphore.h>
#define GERAR_LOGS
#define GERAR_IMAGENS
// #define MOSTRAR_GRAFICOS
#define PRECISAO 0.05 // em metros
#define LARGURA 15.0 // em metros
#define COMPRIMENTO 15.0 // em metros
```

```
#define TETO(X) ((int)((X)+1.0-1e-5))

#define MESMO_PTOI(P1,P2) (P1.x == P2.x && P1.y == P2.y)

#define MAXX (TETO(LARGURA / PRECISAO))

#define MAXY (TETO(COMPRIMENTO / PRECISAO))

#define VALIDO_PTOI(X,Y) (X >= 0 && X < maxx && Y >= 0 && Y < maxy)

typedef unsigned char byte;

typedef struct

{

float x,y;

} ptof_t;

typedef struct

{

int x,y;

} ptoi_t;

// variaveis a serem compartilhadas entre todos os processos

extern byte mapa[MAXX][MAXY];

extern ptof_t traj[MAXX*MAXY]; // traj do GT pro FT

extern ptof_t traj_ctrl[MAXX*MAXY]; // traj do FT pro CTRL

extern uint tamtraj; // nro de ptos da trajetoria armazenada em traj

extern uint tamtraj_ctrl; // nro de ptos da trajetoria armazenada em

traj_ctrl

extern uint itraj; // posic atual; ult posic fornecida ao
controlador

extern ptof_t origem, // pto de inicio de cada trajetoria gerada
trajetoria

(muda!)

origem_global, // pto inicial do robo em todo o processo

destino; // pto de destino da trajetoria (objetivo)

extern int impossivel_gerar_traj;

extern int alcancou_obj;
```

```
extern int ler_toda_volta;

extern float raio, // raio do robo

veloc,

ang_inic;

extern ptof_t posicao; // posicao atual do robo; por enq eh
atualizada pelo sim

extern float direcao; // direcao atual do robo; por enq eh
atualizada pelo sim

#define ALCANCE_SONAR 1.5 // 4.0 metros

#define ANG_PASSO_MOTOR_SONAR 1.8 // graus

#define TEMPO_LEITURA_SONAR 0.025 // segundos; tempo necessario para
o som ir

e voltar 4m (ALCANCE_SONAR)

#define TEMPO_MOV_PASSO_MOTOR_SONAR 0.001 // segundos

#define AMPLITUDE_LEITURA_NORMAL_SONAR 95.0 // graus

#define DIST_CENTRO_SONAR 0.20 // metros

typedef struct

{

ptof_t pos;

float ang,dist;

} dados_sonar_t;

extern dados_sonar_t list_obst[210]; // uma folguinha nao faz mal
pra ninguem :)

extern int tam_list_obst;

extern char nome_arq_amb_real[30]; /* nome do arquivo que contem o
ambiente real,

para fins de simulacao */

extern unsigned char fig[MAXY][MAXX]; /* para gerar arqgif */

extern int maxx;

extern int maxy;
```

```
extern pthread_t thread_gt, thread_ft, thread_do, thread_sonar,
thread_ctrl;

extern sem_t sem_gt, sem_ft, sem_do, sem_sonar, sem_princ, sem_ctrl,
sem_traj_ctrl;

#define OBSTACULO 127 // valor colocado nos bytes correspondentes a
obstaculos

#define MARC_MASC 0x80 // mascara para indicar que pto jah foi
visitado na

busca

#define DESMARC_MASC 0x7F // mascara para desmarcar os ptos do mapa

#define INFINITO 1000.0

#endif

/*****************************************************************
********

var.h - Arquivo de declaração das variáveis globais. Incluido pelo

arquivo celldec.c

******************************************************************
*******/

#include "defs.h"

#include <pthread.h>

#include <semaphore.h>

// variaveis a serem compartilhadas entre todos os processos

byte mapa[MAXX][MAXY];

ptof_t traj[MAXX*MAXY]; // traj do GT pro FT

ptof_t traj_ctrl[MAXX*MAXY]; // traj do FT pro CTRL

uint tamtraj; // nro de ptos da trajetoria armazenada em traj

uint tamtraj_ctrl; // nro de ptos da trajetoria armazenada em
traj_ctrl

uint itraj; // posic atual; ult posic fornecida ao controlador

ptof_t origem, // pto de inicio de cada trajetoria gerada trajetoria
(muda!)

origem_global, // pto inicial do robo em todo o processo
```

```
destino; // pto de destino da trajetoria (objetivo)

int impossivel_gerar_traj;

int alcancou_obj;

int ler_toda_volta;

float raio, // raio do robo

veloc,

ang_inic;

ptof_t posicao; // posicao atual do robo; por enq eh atualizada pelo
sim

float direcao; // direcao atual do robo; por enq eh atualizada pelo
sim

dados_sonar_t list_obst[210]; // uma folguinha nao faz mal pra
ninguem :)

int tam_list_obst;

char nome_arq_amb_real[30]; /* nome do arquivo que contem o ambiente
real,

para fins de simulacao */

unsigned char fig[MAXY][MAXX]; /* para gerar arqgif */

int maxx = MAXX;

int maxy = MAXY;

pthread_t thread_gt, thread_ft, thread_do, thread_sonar,
thread_ctrl;

sem_t sem_gt, sem_ft, sem_do, sem_sonar, sem_princ, sem_ctrl,
sem_traj_ctrl;

/******************************************************************
********

geometria.h - Arquivo de definicoes de tipos e constantes,

e prototipos de funcoes geometricas

*******************************************************************
*******/

#ifndef __GEOMETRIA_H

#define __GEOMETRIA_H
```

```
#define F_IGUAL(X,Y) (fabs((X)-(Y)) < 1e-3)

#define F_MENORIGUAL(X,Y) (((X) < (Y)) || F_IGUAL(X,Y))

#define F_MAIORIGUAL(X,Y) (((X) > (Y)) || F_IGUAL(X,Y))

#define PTO_IGUAL(P1,P2) (F_IGUAL((P1).x,(P2).x) &&
F_IGUAL((P1).y,(P2).y))

#define SQR(X) ((X)*(X))

#define DIST(P1,P2) (sqrt( SQR((P1).x - (P2).x) + SQR((P1).y -
(P2).y) ))

#define GRAUS_PARA_RAD(X) ((X)*M_PI/180.0)

#define RAD_PARA_GRAUS(X) ((X)*180.0/M_PI)

// codigos de retorno da func intersec_retas(...)

#define CONCORRENTES 0

#define PARALELAS -1

#define MESMARETA -2

typedef struct PTO // ED para armazenar um ponto no plano

{

float x,y;

} Pto;

typedef Pto Vetor; // a ED de um vetor eh a mesma de um ponto

extern Vetor vetor(Pto orig, Pto dest);

extern float modulo(Vetor v);

extern Vetor mult_vet_esc(Vetor v, float e);

extern Vetor soma_vet(Vetor v1, Vetor v2);

extern float prod_esc(Vetor v1, Vetor v2);

extern float prod_vet(Vetor v1, Vetor v2);

extern float angulo(Vetor v1, Vetor v2); // retorna o angulo em
radianos entre os

vetores

extern Vetor rot90(Vetor v); // rotaciona v em 90 graus

extern Vetor rot_90(Vetor v); // rotaciona v em -90 graus
```

```
extern Vetor rot(Vetor r, float ang); // rotaciona v em ang radianos

extern int pert_pto_reta(Pto a, Pto b, Pto c); // verifica se os 3
ptos sao colineares

extern int pert_pto_seg(Pto a, Pto b, Pto c);

extern int intersec_retas(Pto p1, Pto p2, Pto q1, Pto q2, Pto * pi);

// *pi: pto de interseccao, se concorrentes

extern int intersec_segmtos(Pto a, Pto b, Pto p, Pto q, Pto * pi1,
Pto * pi2);

// retorna o nro de parametros de retorno utilizados (pi1, pi2)

// se concorrentes -> retorna em pi1 o pto comum

// se mais de um pto em comum -> retorna em pi1 e pi2 os extremos do
segmento comum

#endif

/*******************************************************************
********

celldec.c - Arquivo Principal (Metodo Cell Decomposition)

********************************************************************
*******/

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <math.h>

#include <vga.h>

#include <unistd.h>

#include "defs.h"

#include "var.h"

#include "geometria.h"

#include <pthread.h>

#include <semaphore.h>

extern void detect_obst(void);

extern void ft(void);
```

```
extern void gt(void);

extern void le_sonar(void);

extern void controlador(void);

void sai_sintaxe()

{

printf("Sintaxe:\n\tcelldec -p <arq_param> -r <arq_amb_real> [ -a
<arq_descr_amb> | -s

<arq_amb_salvo> ]\n\n");

exit(1);

}

void erro_sintaxe()

{

printf("\nErro de sintaxe.\n");

sai_sintaxe();

}

void erro(char * msgerro)

{

printf("\n%s\n",msgerro);

exit(1);

}

void erro_abrir_arq(char * nomearq)

{

printf("\nErro ao abrir arquivo %s\n",nomearq);

sai_sintaxe();

}

void carrega_parametros(char * nomearq)

{

FILE * arqpar;

if ((arqpar = fopen(nomearq,"rt")) == NULL)
```

```
erro_abrir_arq(nomearq);

fscanf(arqpar,"%f %f %f %f %f %f

%f",&raio,&veloc,&ang_inic,&(origem.x),&(origem.y),&(destino.x),&(de
stino.y));

origem_global = origem;

direcao = ang_inic;

fclose(arqpar);

}

ptoi_t pto_mapa(ptof_t ptof)

{

ptoi_t ptoi;

ptoi.x = (uint)(ptof.x/PRECISAO);

ptoi.y = (uint)(ptof.y/PRECISAO);

return(ptoi);

}

void marca_pto(ptoi_t p)

{

int raioi=((int)ceil(raio/PRECISAO));

int x,y;

int xi = p.x-raioi >= 0 ? p.x-raioi : 0;

int xf = p.x+raioi < maxx ? p.x+raioi : maxx-1;

int yi = p.y-raioi >= 0 ? p.y-raioi : 0;

int yf = p.y+raioi < maxy ? p.y+raioi : maxy-1;

for (x=xi; x <= xf; x++)

for (y=yi; y <= yf; y++)

mapa[x][y] = OBSTACULO;

}

void marca_linhaX(ptoi_t p1, ptoi_t p2)

{
```

```
uint passo=(int)ceil(raio/PRECISAO),cont=passo;
int s=0;
int dx = p2.x - p1.x;
int dy = p2.y - p1.y;
int yincr = dy > 0? 1 : -1;
int dxdiv2 = dx >> 1;
ptoi_t p = p1;
dy = abs(dy);
marca_pto(p);
while (p.x < p2.x)
{
p.x++;
s += dy;
if (s > dxdiv2)
{
s -= dx;
p.y += yincr;
}
if (cont == passo)
{
cont=1;
marca_pto(p);
}
else
cont++;
}
if (cont > 1) /* se ultimo ponto marcado nao foi o final entao marca
ele */
marca_pto(p);
```

```
}
void marca_linhaY(ptoi_t p1, ptoi_t p2)
{
int s=0;
int dx = p2.x - p1.x;
int dy = p2.y - p1.y;
int xincr = dx > 0? 1 : -1;
int dydiv2 = dy >> 1;
ptoi_t p = p1;
dx = abs(dx);
marca_pto(p);
while (p.y < p2.y)
{
p.y++;
s += dx;
if (s > dydiv2)
{
s -= dy;
p.x += xincr;
}
marca_pto(p);
}
}
void marca_linha(ptoi_t p1, ptoi_t p2)
{
int dx = abs(p2.x - p1.x);
int dy = abs(p2.y - p1.y);
if (dx > dy)
if (p1.x < p2.x)
```

```
marca_linhaX(p1,p2);
else
marca_linhaX(p2,p1);
else
if (p1.y < p2.y)
marca_linhaY(p1,p2);
else
marca_linhaY(p2,p1);
}
void carrega_descr_amb(char * nomearq)
{
int nobsts,o,nverts,v;
ptof_t pf,pinicf;
ptoi_t pi,pinici,panti;
FILE * arqda;
if ((arqda = fopen(nomearq,"rt")) == NULL)
erro_abrir_arq(nomearq);
fscanf(arqda,"%d",&nobsts);
for (o=0; o < nobsts; o++)
{
fscanf(arqda,"%d %f %f",&nverts,&pinicf.x,&pinicf.y);
panti = pinici = pto_mapa(pinicf);
for (v=1; v < nverts; v++, panti=pi)
{
fscanf(arqda,"%f %f",&pf.x,&pf.y);
pi = pto_mapa(pf);
marca_linha(panti,pi);
}
marca_linha(pi,pinici);
```

```
}
fclose(arqda);
}
void carrega_amb_salvo(char * nomearq)
{
FILE * arqas;
if ((arqas = fopen(nomearq,"rb")) == NULL)
erro_abrir_arq(nomearq);
fread(mapa,sizeof(mapa),1,arqas);
fclose(arqas);
}
int main(int argc, char * argv[])
{
pthread_attr_t do_attr, gt_attr, ft_attr, ctrl_attr, sonar_attr;
struct sched_param do_sched_param, gt_sched_param, ft_sched_param,
ctrl_sched_param,
sonar_sched_param;
int descr_amb=0, amb_salvo=0, i;
if (argc == 1)
sai_sintaxe();
if (argc < 3)
erro_sintaxe();
memset(mapa,0,sizeof(mapa));
itraj = 0;
nome_arq_amb_real[0] = '\0';
for (i=1; i < argc; i+=2)
{
if (i == argc-1)
erro_sintaxe();
```

```
if (strcmp(argv[i],"-p") == 0)
carrega_parametros(argv[i+1]);
else
if (strcmp(argv[i],"-a") == 0)
{
if (descr_amb || amb_salvo)
erro_sintaxe();
descr_amb = 1;
carrega_descr_amb(argv[i+1]);
}
else
{
if (strcmp(argv[i],"-s") == 0)
{
if (descr_amb || amb_salvo)
erro_sintaxe();
amb_salvo = 1;
carrega_amb_salvo(argv[i+1]);
}
else
if (strcmp(argv[i],"-r") == 0)
strcpy(nome_arq_amb_real,argv[i+1]);
else
erro_sintaxe();
}
}
setvbuf(stdout,NULL,_IONBF,0);
impossivel_gerar_traj = 0;
alcancou_obj = 0;
```

```
ler_toda_volta = 0;
printf("Vou inic os semaforos\n");
sem_init(&sem_princ,0,0);
sem_init(&sem_gt,0,0);
sem_init(&sem_ft,0,0);
sem_init(&sem_do,0,0);
sem_init(&sem_sonar,0,0);
sem_init(&sem_ctrl,0,0);
sem_init(&sem_traj_ctrl,0,1);
printf("Vou criar os threads\n");
pthread_attr_init(&do_attr);
pthread_attr_setschedpolicy(&do_attr,SCHED_RR);
pthread_attr_setdetachstate(&do_attr,PTHREAD_CREATE_DETACHED);
do_sched_param.sched_priority=4;
pthread_attr_setschedparam(&do_attr,&do_sched_param);
pthread_attr_init(&gt_attr);
pthread_attr_setschedpolicy(&gt_attr,SCHED_RR);
pthread_attr_setdetachstate(&gt_attr,PTHREAD_CREATE_DETACHED);
gt_sched_param.sched_priority=4;
pthread_attr_setschedparam(&gt_attr,&gt_sched_param);
pthread_attr_init(&ft_attr);
pthread_attr_setschedpolicy(&ft_attr,SCHED_RR);
pthread_attr_setdetachstate(&ft_attr,PTHREAD_CREATE_DETACHED);
ft_sched_param.sched_priority=4;
pthread_attr_setschedparam(&ft_attr,&ft_sched_param);
pthread_attr_init(&ctrl_attr);
pthread_attr_setschedpolicy(&ctrl_attr,SCHED_RR);
pthread_attr_setdetachstate(&ctrl_attr,PTHREAD_CREATE_DETACHED);
ctrl_sched_param.sched_priority=0;
```

```
pthread_attr_setschedparam(&ctrl_attr,&ctrl_sched_param);

if (pthread_create(&thread_gt,&gt_attr,(void *(*)(void *))gt,NULL)
!= 0)

printf("Erro criando thread 1\n");

if (pthread_create(&thread_ft,&ft_attr,(void *(*)(void *))ft,NULL)
!= 0)

printf("Erro criando thread 2\n");

if (pthread_create(&thread_do,&do_attr,(void *(*)(void
*))detect_obst,NULL) != 0)

printf("Erro criando thread 3\n");

if (pthread_create(&thread_sonar,&sonar_attr,(void *(*)(void
*))le_sonar,NULL) != 0)

printf("Erro criando thread 4\n");

if (pthread_create(&thread_ctrl,&ctrl_attr,(void *(*)(void
*))controlador,NULL) != 0)

printf("Erro criando thread 5\n");

printf("Vou esperar pelos UPs\n");

sem_wait(&sem_princ);

printf("Recebi um UP\n");

sem_wait(&sem_princ);

printf("Recebi um UP\n");

sem_wait(&sem_princ);

printf("Recebi um UP\n");

sem_wait(&sem_princ);

printf("Recebi um UP\n");

sem_wait(&sem_princ);

printf("Recebi um UP\n");

printf("Principal terminado!\n");

sem_destroy(&sem_princ);

sem_destroy(&sem_gt);

sem_destroy(&sem_ft);
```

```
sem_destroy(&sem_do);

sem_destroy(&sem_sonar);

sem_destroy(&sem_ctrl);

sem_destroy(&sem_traj_ctrl);

return(0);

}

/*******************************************************************
********

gt.c - Arquivo do Gerador de Trajetorias

********************************************************************
*******/

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <math.h>

#include <vga.h>

#include <unistd.h>

#include "defs.h"

#include "geometria.h"

#include <pthread.h>

#include <semaphore.h>

extern ptoi_t pto_mapa(ptof_t ptof);

FILE * gtlog;

/****************** ROTINAS GRAFICAS BASICAS
**********************/

void inicializa_modografico()

{

#ifdef MOSTRAR_GRAFICOS

fprintf(gtlog,"Vou inic o modo grafico\n");

if (vga_init() != 0)
```

```
{
printf("Erro no vga_init()\n");
exit(1);
}
vga_setmode(G640x480x16);
fprintf(gtlog,"Inicializei o modo grafico\n");
#endif
}
void finaliza_modografico()
{
#ifdef MOSTRAR_GRAFICOS
vga_setmode(TEXT);
#endif
}
/*****************************************************************
*/
void desmarca_mapa()
{
int x,y;
for (x=0; x < maxx; x++)
for (y=0; y < maxy; y++)
mapa[x][y] &= DESMARC_MASC;
}
int gera_trajetoria()
{
struct
{
ptoi_t p;
int iant;
```

```
} fila[MAXX*MAXY];

uint ifila=0,ffila=1,i;

int x,y;

int dx[4] = {0,1,0,-1};

int dy[4] = {1,0,-1,0};

ptoi_t desti = pto_mapa(destino);

ptof_t auxptof;

fprintf(gtlog,"Vou gerar uma trajetoria: (%0.2f,%0.2f) ->

(%0.2f,%0.2f)\n",origem.x,origem.y,destino.x,destino.y);

fila[0].p = pto_mapa(origem);

fprintf(gtlog,"fila[0]: (%d,%d) desti:

(%d,%d)\n",fila[0].p.x,fila[0].p.y,desti.x,desti.y);

fila[0].iant = -1;

mapa[fila[0].p.x][fila[0].p.y] = 0;

while (!MESMO_PTOI(fila[ifila].p,desti) && ifila < ffila)

{

/* fprintf(gtlog,"Prim da Fila:
(%d,%d)\n",fila[ifila].p.x,fila[ifila].p.y); */

for (i=0; i < 4; i++)

{

x = fila[ifila].p.x+dx[i];

y = fila[ifila].p.y+dy[i];

if (VALIDO_PTOI(x,y) && (mapa[x][y] == 0))

{

mapa[x][y] |= MARC_MASC;

fila[ffila].p.x = x;

fila[ffila].p.y = y;

fila[ffila].iant = ifila;

ffila++;
```

```
}
}
for (i=0; i < 4; i++)
{
x = fila[ifila].p.x+dx[i]+dx[(i+1)%4];
y = fila[ifila].p.y+dy[i]+dy[(i+1)%4];
if (VALIDO_PTOI(x,y) && (mapa[x][y] == 0))
{
mapa[x][y] |= MARC_MASC;
fila[ffila].p.x = x;
fila[ffila].p.y = y;
fila[ffila].iant = ifila;
ffila++;
}
}
ifila++;
}
if (MESMO_PTOI(fila[ifila].p,desti)) /* encontrou um caminho...*/
{
fprintf(gtlog,"Encontrei caminho!!!\nTrajetoria invertida:
(%d,%d)\n",fila[ifila].p.x,fila[ifila].p.y);
tamtraj = 1;
i = ifila;
traj[0].x = fila[i].p.x*PRECISAO + PRECISAO/2;
traj[0].y = fila[i].p.y*PRECISAO + PRECISAO/2;
while (fila[i].iant != -1)
{
i = fila[i].iant;
traj[tamtraj].x = fila[i].p.x*PRECISAO + PRECISAO/2;
```

```
traj[tamtraj].y = fila[i].p.y*PRECISAO + PRECISAO/2;
tamtraj++;
fprintf(gtlog,"(%d,%d)\n",fila[i].p.x,fila[i].p.y);
}
for (i=0; i < (tamtraj)/2; i++)
{
auxptof = traj[i];
traj[i] = traj[(tamtraj)-i-1];
traj[(tamtraj)-i-1] = auxptof;
}
}
else
{
printf("\n\n\n\n\n\n\nNao ha' caminho unindo origem e destino!\n");
return(0);
}
fprintf(gtlog,"GeraTrajet: Terminei. Vou desmarcar o mapa\n");
desmarca_mapa();
return(1);
}
void mostra_ambiente()
{
int x,y;
#ifdef MOSTRAR_GRAFICOS
vga_setcolor(7);
#endif
for (y=0; y < maxy; y++)
for (x=0; x < maxx; x++)
{
```

```
if (mapa[x][y] == OBSTACULO)
{
#ifdef MOSTRAR_GRAFICOS
vga_setcolor(7);
#endif
fig[y][x] = 150;
}
else
{
#ifdef MOSTRAR_GRAFICOS
vga_setcolor(0);
#endif
}
#ifdef MOSTRAR_GRAFICOS
vga_drawpixel(100+x,100+y);
#endif
}
}
#define COR_TRAJETORIA 15
void mostra_origdest()
{
ptoi_t ptoi;
ptoi = pto_mapa(origem);
#ifdef MOSTRAR_GRAFICOS
vga_setcolor(11);
vga_drawpixel(100+ptoi.x,100+ptoi.y);
vga_drawline(100+ptoi.x-5,100+ptoi.y-5,100+ptoi.x+5,100+ptoi.y+5);
vga_drawline(100+ptoi.x-5,100+ptoi.y+5,100+ptoi.x+5,100+ptoi.y-5);
#endif
```

```
fig[ptoi.y][ptoi.x] = 0;
ptoi = pto_mapa(destino);
fig[ptoi.y][ptoi.x] = 20;
#ifdef MOSTRAR_GRAFICOS
vga_setcolor(13);
vga_drawpixel(100+ptoi.x,100+ptoi.y);
vga_drawline(100+ptoi.x-5,100+ptoi.y-5,100+ptoi.x+5,100+ptoi.y+5);
vga_drawline(100+ptoi.x-5,100+ptoi.y+5,100+ptoi.x+5,100+ptoi.y-5);
#endif
}
void mostra_trajetoria()
{
int i;
ptoi_t ptoi;
#ifdef MOSTRAR_GRAFICOS
vga_setcolor(COR_TRAJETORIA);
#endif
for (i=0; i < tamtraj; i++)
{
ptoi = pto_mapa(traj[i]);
#ifdef MOSTRAR_GRAFICOS
vga_drawpixel(100+ptoi.x,100+ptoi.y);
#endif
fig[ptoi.y][ptoi.x] = 50;
}
}
int visivel_linhaX(ptoi_t p1, ptoi_t p2)
{
uint passo=(int)ceil(raio/PRECISAO),cont=passo;
```

```
int s=0;
int dx = p2.x - p1.x;
int dy = p2.y - p1.y;
int yincr = dy > 0? 1 : -1;
int dxdiv2 = dx >> 1;
ptoi_t p = p1;
dy = abs(dy);
if (mapa[p.x][p.y] == OBSTACULO)
return(0);
while (p.x < p2.x)
{
p.x++;
s += dy;
if (s > dxdiv2)
{
s -= dx;
p.y += yincr;
}
if (cont == passo)
{
cont=1;
if (mapa[p.x][p.y] == OBSTACULO)
return(0);
}
else
cont++;
}
if (cont > 1) /* se ultimo ponto marcado nao foi o final entao marca
ele */
```

```
if (mapa[p.x][p.y] == OBSTACULO)
return(0);
return(1);
}
int visivel_linhaY(ptoi_t p1, ptoi_t p2)
{
int s=0;
int dx = p2.x - p1.x;
int dy = p2.y - p1.y;
int xincr = dx > 0? 1 : -1;
int dydiv2 = dy >> 1;
ptoi_t p = p1;
dx = abs(dx);
if (mapa[p.x][p.y] == OBSTACULO)
return(0);
while (p.y < p2.y)
{
p.y++;
s += dx;
if (s > dydiv2)
{
s -= dy;
p.x += xincr;
}
if (mapa[p.x][p.y] == OBSTACULO)
return(0);
}
}
int visivel_linha(ptoi_t p1, ptoi_t p2)
```

```
{
int dx = abs(p2.x - p1.x);
int dy = abs(p2.y - p1.y);
if (dx > dy)
if (p1.x < p2.x)
return(visivel_linhaX(p1,p2));
else
return(visivel_linhaX(p2,p1));
else
if (p1.y < p2.y)
return(visivel_linhaY(p1,p2));
else
return(visivel_linhaY(p2,p1));
}
void gt_termina()
{
fclose(gtlog);
sem_post(&sem_princ);
}
void _vga_getch(void)
{
#ifdef MOSTRAR_GRAFICOS
vga_getch();
#endif
}
float transf_ang(float ang)
// transforma um ang de 0..2*PI para -PI..PI
{
return(ang <= M_PI? ang : ang - 2*M_PI);
```

```
}
int traj_fora_faixa_leit()
{
int i;
Vetor vet_dir;
Pto _posic,_ptotraj;
_posic.x = posicao.x;
_posic.y = posicao.y;
vet_dir.x = cos(GRAUS_PARA_RAD(direcao));
vet_dir.y = sin(GRAUS_PARA_RAD(direcao));
for (i=1; i < tamtraj; i++)
{
_ptotraj.x = traj[i].x;
_ptotraj.y = traj[i].y;
if (fabs(transf_ang(angulo(vet_dir,vetor(_posic,_ptotraj)))) >
GRAUS_PARA_RAD(AMPLITUDE_LEITURA_NORMAL_SONAR))
{
fprintf(gtlog,"ACHOU TRAJ FORA DA FAIXA: posicao = (%.2f,%.2f)
direcao = %.2f
graus\n",posicao.x,posicao.y,direcao);
fprintf(gtlog," Ponto = (%.2f,%.2f)\n",_ptotraj.x,_ptotraj.y);
fprintf(gtlog," vet_dir = (%.2f,%.2f) angulo = %.2f rad ang_lim =
%.2f\n\n",vet_dir.x,vet_dir.y,fabs(angulo(vet_dir,vetor(_posic,_ptot
raj))),GRAUS_PARA_RA
D(AMPLITUDE_LEITURA_NORMAL_SONAR));
return(1);
}
}
return(0);
}
```

```
void gt(void) /* gerador de trajetorias */
{
int x,y;
int gerou_traj;
int cont, contptos;
FILE * arqfig;
char nomearq[30], s[30];
unsigned char rmap[256], gmap[256], bmap[256];
ptoi_t orig, dest;
for (cont = 0; cont < 256; cont++)
{
rmap[cont] = gmap[cont] = bmap[cont] = cont;
}
cont = 1;
#ifdef GERAR_LOGS
gtlog = fopen("gt.log","wt");
#else
gtlog = fopen("/dev/null","wt");
#endif
setvbuf(gtlog,NULL,_IONBF,0);
inicializa_modografico();
fprintf(gtlog,"maxx=%d maxy=%d\n",maxx,maxy);
while (1)
{
sem_wait(&sem_gt);
fprintf(gtlog,"ORIGEM = (%.2f,%.2f) DESTINO =
(%0.2f,%0.2f)\n",origem.x,origem.y,destino.x,destino.y);
/* mostra mapa*/
orig = pto_mapa(origem);
```

```
dest = pto_mapa(destino);
for (y=0; y < maxy; y++)
{
for (x=0; x < maxx; x++)
if (x == orig.x && y == orig.y)
if (mapa[x][y] == OBSTACULO)
fprintf(gtlog,"M");
else
fprintf(gtlog,"O");
else
if (x == dest.x && y == dest.y)
fprintf(gtlog,"D");
else
fprintf(gtlog,"%c",mapa[x][y] == OBSTACULO? 'X' : ' ');
fprintf(gtlog,"\n");
}
if (!alcancou_obj)
{
gerou_traj = gera_trajetoria();
#ifdef GERAR_IMAGENS
strcpy(nomearq,"fig");
sprintf(s,"%d",cont++);
strcat(nomearq,s);
strcat(nomearq,".gif");
arqfig = fopen(nomearq,"wb");
memset(fig,255,sizeof(fig));
#else
arqfig = fopen("/dev/null","wb");
#endif
```

```
mostra_ambiente();

_vga_getch();

mostra_trajetoria();

mostra_origdest();

_vga_getch();

writegif(arqfig,fig,maxx,maxy,rmap,gmap,bmap,256,1,"");

fclose(arqfig);

if (gerou_traj)

{

for (contptos=0; (contptos < ALCANCE_SONAR/(1.4142*PRECISAO)) &&
(contptos <

tamtraj); contptos++)

{

if (!visivel_linha(pto_mapa(traj[0]),pto_mapa(traj[contptos]))) //
verifica

se traj[cont] eh visivel a partir de traj[0]

break;

}

tamtraj = contptos;

fprintf(gtlog,"NOVO TAMTRAJ = %d\n",tamtraj);

if (tamtraj < 2)

{

fprintf(gtlog,"Deu merda no visivel_linha! tamtraj = %d\n",tamtraj);

}

}

else

{

impossivel_gerar_traj = 1;

fprintf(gtlog,"Nao consegui gerar uma trajetoria\n");

mostra_ambiente();
```

```
_vga_getch();

fprintf(gtlog,"Vou mostrar orig e dest\n");

mostra_origdest();

_vga_getch();

}

}

if (!alcancou_obj && !impossivel_gerar_traj &&
traj_fora_faixa_leit() &&

!ler_toda_volta)

{

fprintf(gtlog,"Eh preciso ler toda a volta!\n");

ler_toda_volta = 1;

sem_post(&sem_do);

}

else // soh se nao precisar fazer leitura total do sonar

{

ler_toda_volta = 0;

fprintf(gtlog,"\n ENVIANDO UP PRO FT!!!\n");

sem_post(&sem_ft);

}

if (alcancou_obj)

break;

if (impossivel_gerar_traj)

break;

}

fprintf(gtlog,"Sai do laco\n");

if (alcancou_obj)

fprintf(gtlog," Com sucesso\n");

else
```

```
fprintf(gtlog," Sem sucesso\n");
#ifdef MOSTRAR_GRAFICOS
fprintf(gtlog,"Esperando getch para sair\n");
_vga_getch();
finaliza_modografico();
#endif
gt_termina();
}
/*******************************************************************
********
do.c - Arquivo do Detector de Obstaculos
********************************************************************
*******/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <vga.h>
#include <unistd.h>
#include "defs.h"
#include "geometria.h"
#include <pthread.h>
#include <semaphore.h>
extern ptoi_t pto_mapa(ptof_t ptof);
FILE * dolog;
void do_termina()
{
fprintf(dolog,"DO: Vou terminar\n");
fclose(dolog);
sem_post(&sem_princ);
```

```
}
void detect_obst() /* detector de obstaculos */
{
ptof_t pf;
ptoi_t pi, pt, pdest=pto_mapa(destino);
int i, o;
int x,y;
int prim_vez=1;
#ifdef GERAR_LOGS
dolog = fopen("do.log","wt");
#else
dolog = fopen("/dev/null","wt");
#endif
setvbuf(dolog,NULL,_IONBF,0);
fprintf(dolog,"DO: Vou entrar no loop\n");
while (1)
{
sem_post(&sem_sonar);
fprintf(dolog,"DO: Acordei o SIM\n");
if (impossivel_gerar_traj)
break;
if (!alcancou_obj)
{
fprintf(dolog,"DO: Vou esperar pelo SIM\n");
sem_wait(&sem_do); // espera pelo SIM
fprintf(dolog,"DO: Acordei\n");
for (o=0; o < tam_list_obst; o++)
{
fprintf(dolog,"pos = (%.2f,%.2f) ang = %.2f dist =
```

```
%.2f\n",list_obst[o].pos.x,list_obst[o].pos.y,list_obst[o].ang,list_
obst[o].dist);
pi = pto_mapa(list_obst[o].pos);
if (list_obst[o].dist < INFINITO)
{
fprintf(dolog,"Detectou Obstaculo\n");
pf.x = list_obst[o].pos.x +
list_obst[o].dist*cos(GRAUS_PARA_RAD(list_obst[o].ang));
pf.y = list_obst[o].pos.y +
list_obst[o].dist*sin(GRAUS_PARA_RAD(list_obst[o].ang));
fprintf(dolog,"\nSonar enviou detectou obstaculo
(%.4f,%.4f)\n",pf.x,pf.y);
pi = pto_mapa(pf);
marca_pto(pi);
}
}
/* mostra mapa*/
for (y=0; y < maxy; y++)
{
for (x=0; x < maxx; x++)
fprintf(dolog,"%c",mapa[x][y] == OBSTACULO? 'X' : ' ');
fprintf(dolog,"\n");
}
}
fprintf(dolog,"Vou acordar o GT\n");
sem_post(&sem_gt);
if (alcancou_obj)
break;
fprintf(dolog,"Vou esperar pelo FT ou pelo GT\n");
```

```
sem_wait(&sem_do); // espera ft ou gt (q pode pedir para
ler_toda_volta)

if (!ler_toda_volta) // entao quem me acordou foi o ft; vou
atualizar origem

origem = traj[tamtraj-1];

fprintf(dolog,"Testando se objetivo alcancado: origem =

(%.2f,%.2f)\n",origem.x,origem.y);

if (MESMO_PTOI(pto_mapa(origem),pdest))

{

fprintf(dolog,"\nOBJETIVO ALCANCADO!!!\n");

alcancou_obj = 1;

}

}

do_termina();

}

/*******************************************************************
********

ft.c - Arquivo do Fornecedor de Trajetorias

********************************************************************
*******/

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <math.h>

#include <vga.h>

#include <unistd.h>

#include "defs.h"

#include "geometria.h"

#include <pthread.h>

#include <semaphore.h>
```

```
// variaveis no controlador

extern int acordou_ft;

extern double tinic, treal;

FILE * ftlog;

void ft_termina()

{

fprintf(ftlog,"FT: Vou Terminar\n");

fclose(ftlog);

sem_post(&sem_princ);

}

ptof_t pto_ctrl(ptof_t ptraj)

// transforma as coordenadas do sistema de referencia adotado para o
do robo

{

ptof_t pctrl, paux;

float ang = GRAUS_PARA_RAD(ang_inic);

paux.x = ptraj.x - origem_global.x; // translacao p/ posicao
original

paux.y = ptraj.y - origem_global.y;

pctrl.x = cos(ang)*paux.x + sin(ang)*paux.y; // rotacao de ang_inic

pctrl.y = -sin(ang)*paux.x + cos(ang)*paux.y;

fprintf(ftlog,"(%.2f,%.2f) -->
(%.2f,%.2f)\n",ptraj.x,ptraj.y,pctrl.x,pctrl.y);

return(pctrl);

}

/* ERRADA!!!

ptof_t pto_sist_orig(ptof_t pctrl)

// transforma as coordenadas do sistema de referencia do robo para o
adotado

{
```

```
ptof_t ptraj;

ptraj.y = pctrl.x + origem_global.y + raio;

ptraj.x = -(pctrl.y - origem_global.x);

return(ptraj);

}

*/

void atualiza_traj_ctrl(void)

{

int i;

sem_wait(&sem_traj_ctrl);

for (i=0; i < tamtraj; i++)

{

// passando as coordenadas da traj gerada para o sist de coordenadas
do robo

traj_ctrl[i] = pto_ctrl(traj[i]);

}

acordou_ft = 0;

tinic = treal;

tamtraj_ctrl = tamtraj;

sem_post(&sem_traj_ctrl);

}

void ft() /* fornecedor de trajetoria */

{

int ctrl_acordado = 0;

#ifdef GERAR_LOGS

ftlog = fopen("ft.log","wt");

#else

ftlog = fopen("/dev/null","wt");

#endif
```

```
setvbuf(ftlog,NULL,_IONBF,0);

while (1)

{

fprintf(ftlog,"Vou esperar trajetoria\n");

sem_wait(&sem_ft);

if (!alcancou_obj && !impossivel_gerar_traj)

{

fprintf(ftlog,"Recebi trajetoria\n");

atualiza_traj_ctrl();

}

if (!ctrl_acordado)

{

sem_post(&sem_ctrl);

ctrl_acordado = 1;

}

if (!alcancou_obj && !impossivel_gerar_traj)

{

fprintf(ftlog,"Vou esperar pelo controlador\n");

sem_wait(&sem_ft); // espera que o controlador forneca toda a
trajetoria

fprintf(ftlog,"Vou esperar o robo parar\n");

espera_robo_parar(ftlog);

}

sem_post(&sem_do);

if (alcancou_obj || impossivel_gerar_traj)

break;

fprintf(ftlog,"ORIGEM = (%0.2f,%0.2f) DESTINO = (%0.2f,%0.2f)
TAMTRAJ =

%d\n",traj[0].x,traj[0].y,traj[tamtraj-1].x,traj[tamtraj-
1].y,tamtraj);
```

```
}
ft_termina();
}
/*****************************************************************
********
le_sonar.c - Arquivo do thread de leitura do sonar
*****************************************************************
*******/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <unistd.h>
#include "defs.h"
#include "geometria.h"
#include <pthread.h>
#include <semaphore.h>
extern void le_sonar(void);
extern void inic_sonar(void);
extern void fim_sonar(void);
#define RAIO_REAL (raio - 0.10)
FILE * sonarlog;
extern void erro(char *);
extern ptoi_t pto_mapa(ptof_t ptof);
void sonar_termina()
{
fprintf(sonarlog,"SIM: Vou teminar; entrei no sonar_termina\n");
fclose(sonarlog);
sem_post(&sem_princ);
}
void le_sonar()
```

```
{
ptof_t posic;
double ang, a,
delta; /* leitura do sonar vai de -delta ateh +delta */
double dist;
Vetor dir;
ptoi_t pi;
int cont=0, i;
int x,y;
int prim_vez = 1;
ang = 0.0;
#ifdef GERAR_LOGS
sonarlog = fopen("sonar.log","wt");
#else
sonarlog = fopen("/dev/null","wt");
#endif
setvbuf(sonarlog,NULL,_IONBF,0);
inic_sonar();
while (1)
{
sem_wait(&sem_sonar); // solicitacao de leitura do sonar
if (alcancou_obj || impossivel_gerar_traj)
break;
if (prim_vez)
{
prim_vez = 0;
posic = origem; // que serah origem_global
}
else
```

```
{
if (!ler_toda_volta)
posic = traj[tamtraj-1];
// se eh para ler toda a volta, entao nao se moveu e continua na
mesma posicao
}
posicao = posic; // posicao eh global, a ser usada pelas demais
funcoes
// a atualizacao da direcao agora eh feita pelo controlador!
cont = 0;
if (ler_toda_volta)
delta = 180.0;
else
delta = AMPLITUDE_LEITURA_NORMAL_SONAR;
sonar_rpc();
sem_post(&sem_do);
}
fim_sonar();
sonar_termina();
}
/*******************************************************************
********
geometria.c - Arquivo de Funcoes Geometricas
********************************************************************
*******/
#include <math.h>
#include "geometria.h"
Vetor vetor(Pto orig, Pto dest)
{
Vetor v;
```

```
v.x = dest.x - orig.x;
v.y = dest.y - orig.y;
return(v);
}
float modulo(Vetor v)
{
return(sqrt(v.x*v.x + v.y*v.y));
}
Vetor mult_vet_esc(Vetor v, float e)
{
v.x *= e;
v.y *= e;
return(v);
}
Vetor soma_vet(Vetor v1, Vetor v2)
{
Vetor s;
s.x = v1.x + v2.x;
s.y = v1.y + v2.y;
return(s);
}
float prod_esc(Vetor v1, Vetor v2)
{
return(v1.x * v2.x + v1.y * v2.y);
}
float prod_vet(Vetor v1, Vetor v2)
{
return(v1.x * v2.y - v1.y * v2.x);
}
```

```
float angulo(Vetor v1, Vetor v2)

/* retorna o angulo em radianos entre os vetores

o angulo eh medido a partir de v1, em sentido anti-horario ateh v2
*/

{

float modv1 = modulo(v1);

float modv2 = modulo(v2);

float pv;

if (modv1*modv2 == 0.0)

return(0.0);

if ((pv = prod_vet(v1,v2)) == 0.0)

return(prod_esc(v1,v2) > 0.0 ? 0.0 : M_PI);

return(pv > 0.0 ? acos(prod_esc(v1,v2)/(modv1*modv2)) : 2*M_PI -

acos(prod_esc(v1,v2)/(modv1*modv2)));

}

Vetor rot90(Vetor v) // rotaciona v em 90 graus

{

Vetor r;

r.x = -v.y;

r.y = v.x;

return(r);

}

Vetor rot_90(Vetor v) // rotaciona v em -90 graus

{

Vetor r;

r.x = v.y;

r.y = -v.x;

return(r);

}
```

```
Vetor rot(Vetor v, float ang)

{

Vetor r;

r.x = v.x*cos(ang) - v.y*sin(ang);

r.y = v.x*sin(ang) + v.y*cos(ang);

return(r);

}

int pert_pto_reta(Pto a, Pto b, Pto c) // verifica se os 3 ptos sao
colineares

{

return(F_IGUAL(prod_vet(vetor(a,b),vetor(a,c)),0.0));

}

int pert_pto_seg(Pto a, Pto b, Pto c)

{

if (pert_pto_reta(a,b,c))

return( F_MENORIGUAL (prod_esc(vetor(c,a),vetor(c,b)),0.0));

return(0);

}

#define CONCORRENTES 0

#define PARALELAS -1

#define MESMARETA -2

int intersec_retas(Pto p1, Pto p2, Pto q1, Pto q2, Pto * pi)

// *pi: pto de interseccao, se concorrentes

{

Vetor d1, d2;

double k;

if (F_IGUAL(prod_vet(d1 = vetor(p1,p2), d2 = vetor(q1,q2)),0.0))

return(pert_pto_reta(p1,p2,q1)?MESMARETA:PARALELAS);

k = prod_vet(vetor(p1,q1),d1) / prod_vet(d1,d2);
```

```
pi->x = q1.x + k * d2.x;

pi->y = q1.y + k * d2.y;

return(CONCORRENTES);

}

int intersec_segmtos(Pto a, Pto b, Pto p, Pto q, Pto * pi1, Pto *
pi2)

// retorna o nro de parametros de retorno utilizados (pi1, pi2)

// se concorrentes -> retorna em pi1 o pto comum

// se mais de um pto em comum -> retorna em pi1 e pi2 os extremos do
segmento comum

{

switch(intersec_retas(a,b,p,q,pi1))

{

case PARALELAS : return(0);

case CONCORRENTES : return(pert_pto_seg(a,b,*pi1) &&
pert_pto_seg(p,q,*pi1));

case MESMARETA : if (pert_pto_seg(p,q,a))

{

*pi1 = a;

*pi2 = pert_pto_seg(p,q,b)?b:(pert_pto_seg(a,b,p)?p:q);

}

else

if (pert_pto_seg(p,q,b))

{

*pi1 = b;

*pi2 = pert_pto_seg(a,b,p)?p:q;

}

else

if (pert_pto_seg(a,b,p))

{
```

```
*pi1 = p;
*pi2 = q;
}
else
return(0);
return(PTO_IGUAL(*pi1,*pi2)?1:2);
}
return(0); // nunca deve chegar aqui!
}
/*****************************************************************
********
sonar.cpp - Arquivo de interface e leitura efetiva dos sonares,
adaptado a partir de programa desenvolvido por Felipe Renon
******************************************************************
*******/
#include <iostream.h>
#include <gif.h>
#include<unistd.h>
#include <conio.h>
#include <ciostream.h>
#include<iostream.h>
#include<delay.h>
#include<stdlib.h>
#include<stdio.h>
#include<math.h>
#include <time.h>
#include <sys/wait.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
```

```
#include <twldrive.h>

#include "defs.h"

#include "geometria.h"

#define DIST_MIN 0.26

#define DIST_MAX 6.0 // em metros; distancia maxima que espera-se
que

// detecte um obstaculo; se detectar algo depois

// disto, atrapalharah a leitura seguinte

#define DELAY ((int)(2*1000*DIST_MAX/343.2))// delay em ms referente
a DIST_MAX

float delta,_angulo;

#define PRECISION 0.50

#define PADROES 4

#define ARC 2*PI

#define RED 254

#define GREEN 253

#define BLUE 252

#define PEN 3

extern ROVER_DRIVER rov;

extern FILE * sonarlog;

void bline(int x1, int y1, int x2, int y2, char color, unsigned char
pic[])

{

/* Desenha uma linha (x1,y1)-(x2,y2) de cor 'color' no bitmap 'pic'
*/

short int dx, dy, sdx, sdy, x, y, px, py;

dx = x2 - x1;

dy = y2 - y1;

sdx = (dx < 0) ? -1 : 1;

sdy = (dy < 0) ? -1 : 1;
```

```
dx = sdx * dx + 1;
dy = sdy * dy + 1;
x = y = 0;
px = x1;
py = y1;
if (dx >= dy) {
for (x = 0; x < dx; x++) {
pic[py*640 + px] = color;
y += dy;
if (y >= dx) {
y -= dx;
py += sdy;
}
px += sdx;
}
} else {
for (y = 0; y < dy; y++) {
pic[py*640 + px] = color;
x += dx;
if (x >= dy) {
x -= dy;
px += sdx;
}
py += sdy;
}
}
}
void circle(int x, int y, int raio, unsigned char pic[])
{
```

```
/* Circulo de centro x,y e raio 'raio' no bitmap 'pic' */
int cor=BLUE;
int w=640;
int xoff=0;
int yoff=raio;
int balance= -raio;
do
{
*(pic+(y+yoff)*w+(x+xoff))=cor;
*(pic+(y+yoff)*w+(x-xoff))=cor;
*(pic+(y-yoff)*w+(x-xoff))=cor;
*(pic+(y-yoff)*w+(x+xoff))=cor;
*(pic+(y+xoff)*w+(x-yoff))=cor;
*(pic+(y+xoff)*w+(x+yoff))=cor;
*(pic+(y-xoff)*w+(x+yoff))=cor;
*(pic+(y-xoff)*w+(x-yoff))=cor;
if ((balance += xoff++ + xoff) >= 0)
balance -= --yoff + yoff;
} while (xoff <= yoff);
}
void plot_point(int x, int y, char pic[])
{
/* Ponto simples x,y no bitmap 'pic' */
*(pic+y*640+x)=0;
}
extern "C" void inic_sonar(void);
void inic_sonar(void)
{
rov.on();
```

```
}
extern "C" void fim_sonar(void);
void fim_sonar(void)
{
// rov.off(); quem desligara sera o controlador
}
extern "C" void sonar_rpc(void);
void sonar_rpc(void)
{
char nome_arq_fig[20] = "figsonar.gif";
/* Tamanho da imagem */
const int w=640;
const int h=480;
/* Constantes para gerar GIF */
unsigned char *rmap=new unsigned char[256];
unsigned char *gmap=new unsigned char[256];
unsigned char *bmap=new unsigned char[256];
const int numcols=256;
const int colorstyle=0;
char *comment="Descricao de RPCs";
FILE *fp=fopen(nome_arq_fig,"wb");
if(!fp)
{
printf("Can't open output file:%s\n",nome_arq_fig);
return;
}
unsigned char *pic=new unsigned char [w*h];
if(!pic)
{
```

```
printf("Can't allocate memory\n");
return;
}
int i;
for(i=0;i < 256;i++)
{
rmap[i]=i;
gmap[i]=i;
bmap[i]=i;
}
rmap[254]=255;
gmap[254]=0;
bmap[254]=0;
rmap[253]=0;
gmap[253]=255;
bmap[253]=0;
rmap[252]=0;
gmap[252]=0;
bmap[252]=255;
int x,y;
/* Apaga (pinta) o bitmap na memoria */
for(y=0;y < h;y++) for(x=0;x < w;x++) *(pic+y*w+x)=255;
float _angulo;
int ang;
float roh[201];
float rpc[201];
/* Circunferencias de referencia */
circle(320,240,50,pic);
circle(320,240,150,pic);
```

```
circle(320,240,100,pic);

circle(320,240,200,pic);

fprintf(sonarlog,"Vou dar reset\n");

cout << "* Inicializando posicao inicial\n";

rov.pmotor.reset();

/* De 0 a 200, gira o pescoco para cobrir 360 graus, colocando no
array

roh[] as distancias */

// ajusta delta

if (ler_toda_volta)

delta = 180.0;

else

delta = AMPLITUDE_LEITURA_NORMAL_SONAR

tam_list_obst = 2*(int)(delta/ANG_PASSO_MOTOR_SONAR);

// posiciona em -delta

fprintf(sonarlog,"Vou para posicao inicial\n");

for (i=0;i<tam_list_obst/2;i++)

{

rov.pmotor.step(0);

delay(10);

}

cout << "* Efetuando varredura\n";

fprintf(sonarlog,"Vou fazer a varredura\n");

for (i=0;i<tam_list_obst;i++)

{

if (ler_toda_volta)

rov.tsonar.trig();

else

rov.bsonar.trig();
```

```
delay(DELAY);
if (ler_toda_volta)
roh[i]=(rov.tsonar.utof()*343.2*1e-6)/2*50;
else
roh[i]=(rov.bsonar.utof()*343.2*1e-6)/2*50;
printf("roh %.2f\n",roh[i]);
// inicializa todas as RPC's com o tamanho do primeiro circulo
rpc[i]=0;
rov.pmotor.step(1);
}
cout << "* Reset \n";
/* Retorna o giro */
for (i=0;i<tam_list_obst/2;i++)
{
rov.pmotor.step(0);
delay(10);
}
rov.pmotor.reset();
/* Gera a imagem a partir de roh[], decompondo as distancias em suas
respectivas componentes XY */
roh[tam_list_obst]=roh[tam_list_obst-1];
/* Analisa as possiveis RPC's */
cout << "* localizando RPC's\n";
float media, n_arc=0, ultimo =0;
int n=0, inic_ang=0, fim_ang=0;
int n_parede=0;
char copy_data=0;
for (int i=0; i<tam_list_obst; i++)
{
```

```
if ((ultimo!=0)&&(fabs(ultimo-roh[i])<=PRECISION))
{
if (n==0) inic_ang=i-1;
n++;
n_arc =n_arc + PI*roh[i-1]/100;
if ((n>=PADROES)||(n_arc>=ARC))
{
copy_data=1;
fim_ang=i;
}
}
else
{
media = roh[inic_ang];
if ((copy_data) || ((i==199)&&(copy_data!=0)))
{
copy_data=0;
n_parede++;
/* calcula a media da RPC */
for (int k=inic_ang; k<=fim_ang; k++) media =
(media+roh[k])/2;
for (int k=inic_ang; k<=fim_ang; k++)
{
rpc[k]= media;
}
}
n=0;
media=0;
n_arc=0;
```

```
}
ultimo=roh[i];
}
media=roh[inic_ang];
if (copy_data)
{
copy_data=0;
n_parede++;
/* calcula a media da RPC */
for (int k=inic_ang; k<=fim_ang; k++) media = (media+roh[k])/2;
for (int k=inic_ang; k<=fim_ang; k++)
{
rpc[k]= media;
}
}
cout << "Total de " << n_parede << " rpc(s) encontrada(s)\n\n";
int x2,y2;
/* tracando linhas entre pontos vizinhos */
for (ang=0;ang<tam_list_obst;ang++)
{
printf(" %.2f\n",rpc[ang]);
_angulo=ang*1.8/180*PI;
x=320+int(rint(roh[ang]*cos(_angulo)));
if (x<0) x=0;
if (x>639) x=639;
y=240-int(rint(roh[ang]*sin(_angulo)));
if (y<0) y=0;
if (y>479) y=479;
_angulo=(ang+1)*1.8/180*PI;
```

```
x2=320+int(rint(roh[ang+1]*cos(_angulo)));
if (x2<0) x2=0;
if (x2>639) x2=639;
y2=240-int(rint(roh[ang+1]*sin(_angulo)));
if (y2<0) y2=0;
if (y2>479) y2=479;
bline(x,y,x2,y2,0,pic);
}
/* tracando linhas entre as RPC's */
for (ang=0;ang<tam_list_obst;ang++)
{
/* Passa filtro */
if ((rpc[ang]!= 0)&&(rpc[ang+1]!=0)&&(rpc[ang]==rpc[ang+1]))
for (i=1; i <= PEN; i++)
{
_angulo=ang*1.8/180*PI;
x=320+int(rint((rpc[ang]+i)*cos(_angulo)));
if (x<0) x=0;
if (x>639) x=639;
y=240-int(rint((rpc[ang]+i)*sin(_angulo)));
if (y<0) y=0;
if (y>479) y=479;
_angulo=(ang+1)*1.8/180*PI;
x2=320+int(rint((rpc[ang+1]+i)*cos(_angulo)));
if (x2<0) x2=0;
if (x2>639) x2=639;
y2=240-int(rint((rpc[ang+1]+i)*sin(_angulo)));
if (y2<0) y2=0;
if (y2>479) y2=479;
```

```
bline(x,y,x2,y2,RED,pic);
}
}
/* Traca vetor da trajetoria */
float maior = rpc[0];
int k1=0, k2=0;
for (i=1; i<tam_list_obst; i++)
{
if (maior <= rpc[i])
{
k1 = i;
if (maior==rpc[i]) k2++;
else k2=0;
maior=rpc[i];
}
}
_angulo=(2*k1-k2)*0.9/180*PI;
x=320+int(rint((maior)*cos(_angulo)));
if (x<0) x=0;
if (x>639) x=639;
y=240-int(rint((maior)*sin(_angulo)));
if (y<0) y=0;
if (y>479) y=479;
_angulo=(ang+1)*1.8/180*PI;
x2=320;
if (x2<0) x2=0;
if (x2>639) x2=639;
y2=240;
if (y2<0) y2=0;
```

```
if (y2>479) y2=479;

bline(x,y,x2,y2,GREEN,pic);

/* Escreve 'pic' para o formato GIF */

// cout << "Gerando arquivo " << argv[1] << "\n";

cout << "\n\nARCO da RPC : " << k2*1.8;

cout << "\nlocalizacao <Dist,ang> : ( " << maior/50 << ", " <<
(2*k1-

k2)*0.9 << ").";

cout << "\nCoordenada cartesiana (x,y) : ( " <<
maior/50*sin(_angulo) << ", "

<< maior/50*cos(_angulo) <<").";

writegif(fp,pic,w,h,rmap,gmap,bmap,numcols,colorstyle,comment);

fclose(fp);

free(pic);

// cria list_obst

_angulo = direcao - delta;

ptof_t pos_sonar;

pos_sonar.x = posicao.x +
DIST_CENTRO_SONAR*cos(GRAUS_PARA_RAD(direcao));

pos_sonar.y = posicao.y +
DIST_CENTRO_SONAR*sin(GRAUS_PARA_RAD(direcao));

for (i=0; i < tam_list_obst; i++)

{

rpc[i] = rpc[i] / 50;

printf("%.2f\n",rpc[i]);

list_obst[i].pos = pos_sonar;

list_obst[i].ang = _angulo;

if (rpc[i] > DIST_MIN && rpc[i] < ALCANCE_SONAR)

list_obst[i].dist = rpc[i];

else
```

```
list_obst[i].dist = INFINITO;
_angulo += ANG_PASSO_MOTOR_SONAR;
// printf("P=(%.2f,%.2f)",list_obst[i].pos.x,list_obst[i].pos.y);
printf("A=%.2f D=%.2f \n",list_obst[i].ang,list_obst[i].dist);
}
}
/*******************************************************************
********
twlckin.cpp - Arquivo do Controlador - Modelo Cinemático - Robô Twil
Adaptado a partir de programa desenvolvido por Walter Lages
********************************************************************
*******/
#define REAL_TIME
#ifdef REAL_TIME
#include <fcntl.h>
#include <unistd.h>
#endif
#include <ciostream.h>
#include <conio.h>
#include <fstream.h>
#include <contain.h>
#include <twldrive.h>
#include <twlparam.h>
#include <twlpath.h>
#include <twltype.h>
#ifdef REAL_TIME
#include <rtc.h>
#endif
#include <control.h>
#include <semaphore.h>
```

```
#include <pthread.h>
#include <delay.h>
#include "geometria.h"
#include "defs.h"
#include <stdio.h>
#define ALPHA0 1
#define ALPHA1 ALPHA0
int t_acom = (int)(1000*(3*2*M_PI/ALPHA0));
ROVER_DRIVER rov;
static const double ST=0.05;
FILE * ctrllog;
// variaveis para fornecimento da trajetoria
int acordou_ft;
double tinic=0.0, treal=0.0;
struct ODOMETRY
{
int npr;
int npl;
double ur;
double ul;
double x;
double y;
double phi;
};
cvector path(double t_atual)
{
cvector p(2);
double t;
uint itraj;
```

```
sem_wait(&sem_traj_ctrl);
t = tinic;
itraj = 0;
while (t < t_atual && itraj < tamtraj_ctrl-1)
{
t += DIST(traj_ctrl[itraj],traj_ctrl[itraj+1])/veloc;
itraj++;
}
p[0] = traj_ctrl[itraj].x;
p[1] = traj_ctrl[itraj].y;
fprintf(ctrllog,"tamtraj_ctrl = %d PATH =
(%.2f,%.2f)\n",tamtraj_ctrl,p[0],p[1]);
if (itraj == tamtraj_ctrl-1 && !acordou_ft)
{
fprintf(ctrllog,"Acordando o FT! T = %.3f\n",t_atual);
sem_post(&sem_ft);
acordou_ft = 1;
}
sem_post(&sem_traj_ctrl);
return p;
}
// vehicle dead-reckoning (center of wheels)
//
// p(t+1)=dr(p(t),u(t))
cvector dr(const cvector &p,const cvector &u)
{
double deltaD=(u[0]+u[1])/2.0;
double deltatheta=(u[0]-u[1])/WHEELBASE;
double deltatheta2=deltatheta/2.0;
```

```
double deltasigma=(deltatheta==0.0)?
deltaD:deltaD*sin(deltatheta2)/deltatheta2;

cvector p1(3);

p1[0]=deltasigma*cos(p[2]+deltatheta2);

p1[1]=deltasigma*sin(p[2]+deltatheta2);

p1[2]=deltatheta;

return p+p1;

}

// vehicle kinematic transform (center of wheels position->center of
mass positon)

//

// cm=cw2cm(cw)

cvector cw2cm(const cvector &cw)

{

const double d=CENTER2WHEEL_AXIS;

cvector cm(3);

cm[0]=cw[0]-d*cos(cw[2]);

cm[1]=cw[1]-d*sin(cw[2]);

cm[2]=cw[2];

return cm;

}

// vehicle kinematic transform (center of mass position->center of
wheels positon)

//

// cw=cm2cw(cm)

cvector cm2cw(const cvector &cm)

{

const double d=CENTER2WHEEL_AXIS;

cvector cw(3);

cw[0]=cm[0]+d*cos(cm[2]);
```

```
cw[1]=cm[1]+d*sin(cm[2]);
cw[2]=cm[2];
return cw;
}
// vehicle system output function
//
// y(t)=h(x)
cvector h(const cvector &x)
{
cvector y(2);
double s=sin(x[2]);
double c=cos(x[2]);
y[0]=x[0]+Xcr*c-Ycr*s;
y[1]=x[1]+Xcr*s+Ycr*c;
return y;
}
// vehicle system output function with orientation
//
// y(t)=ho(q)=ho(x)
cvector ho(const cvector &q)
{
cvector y(3);
double s=sin(q[2]);
double c=cos(q[2]);
y[0]=q[0]+Xcr*c-Ycr*s;
y[1]=q[1]+Xcr*s+Ycr*c;
y[2]=q[2];
return y;
}
```

```
// invbeta(x)
matrix invbeta(const cvector &x)
{
const double c=WHEEL_RADIUS/(2.0*AXIS2WHEEL);
const double b=AXIS2WHEEL;
const double d=CENTER2WHEEL_AXIS;
double sphi=sin(x[2]);
double cphi=cos(x[2]);
matrix B(2,2);
B[0][0]=(d-Xcr)*cphi+(b+Ycr)*sphi;
B[0][1]=-((b+Ycr)*cphi+(-d+Xcr)*sphi);
B[1][0]=-((-d+Xcr)*cphi+(b-Ycr)*sphi);
B[1][1]=(b-Ycr)*cphi+(d-Xcr)*sphi;
return 1/(2.0*c*b*(d-Xcr))*B;
}
// Model reference
// .
// ym(t)=g(x,u)
cvector g(const cvector &x,const cvector &u)
{
cvector G(2);
G[0]=-ALPHA0*x[0]+ALPHA0*u[0];
G[1]=-ALPHA1*x[1]+ALPHA1*u[1];
return G;
}
extern "C" void espera_robo_parar(FILE * arqlog);
void espera_robo_parar(FILE * arqlog)
{
fprintf(arqlog,"delay em miliseg: %d\n",t_acom);
```

```
delay(t_acom);
}
extern "C" void controlador(void);
void controlador(void)
{
PTRLIST<struct ODOMETRY> odolist;
// initial state
cvector x(3);
x[0]=X0;
x[1]=Y0;
x[2]=PHI0;
// odomety initial conditions
cvector xo=cm2cw(x);
// Model reference initial conditions
cvector ym=h(x);
// Reference path
cvector yr(3);
yr[0]=0;
yr[1]=0;
yr[2]=0;
// LINE_PATH path(yr,1,5.5e-2); //ym=y
// LINE_PATH path(yr,2.0,10e-2); //ym=y
cvector rgain(3);
rgain[0]=2.5; // 2.52;
rgain[1]=3.25; // 0.756*3;
rgain[2]=0.05; // 0.189;
cvector lgain=rgain;
double ur=0.0;
double ul=0.0;
```

```
PID_CONTROLLER rpid(rgain[0],rgain[1],rgain[2]);
PID_CONTROLLER lpid(lgain[0],lgain[1],rgain[2]);
rpid.saturate(1,-11.7,11.7);
lpid.saturate(1,-11.7,11.7);
int error=0;
int xs=wherex();
int ys=wherey();
#ifdef GERAR_LOGS
ctrllog = fopen("ctrl.log","wt");
#else
ctrllog = fopen("/dev/null","wt");
#endif
setvbuf(ctrllog,NULL,_IONBF,0);
fprintf(ctrllog,"\nVou aguardar no semaforo sem_ctrl\n");
sem_wait(&sem_ctrl);
#ifdef REAL_TIME
volatile int flag=1;
int timer=open("/dev/rtctimer",O_RDWR);
unsigned long blocktime;
const unsigned long STj=(unsigned long)(ST*1000.0*1024.0/1000+0.5);
write(timer,&STj,sizeof(STj));
#endif
// int nmax=(int)(path.time()/ST*5);
for(int n=0; !alcancou_obj && !impossivel_gerar_traj; n++)
{
double t=n*ST;
treal = t;
#ifdef REAL_TIME
if(n)
```

```
{
if(flag)
{
cout << "\n\nOverrun!\n";
return ;
}
}
read(timer,&blocktime,sizeof(blocktime));
flag=(blocktime <= 1)? 1:0;
write(timer,&STj,sizeof(STj));
#endif
cvector p=path(t);
int npr=rov.rtacho.read();
rov.rtacho.clear();
int npl=rov.ltacho.read();
rov.ltacho.clear();
double rvel=npr*2.0*M_PI/rov.rtacho.PULSES/ST;
double lvel=npl*2.0*M_PI/rov.ltacho.PULSES/ST;
cvector u(2);
u[0]=rvel*WHEEL_RADIUS*ST;
u[1]=lvel*WHEEL_RADIUS*ST;
xo=dr(xo,u);
x=cw2cm(xo);
cvector y=h(x);
cvector coord_robo=ho(x);
direcao = RAD_PARA_GRAUS(coord_robo[2]) + ang_inic;
fprintf(ctrllog,"t=%.3f: (%.2f,%.2f) ang=%.2f
\n",treal,y[0],y[1],direcao);
cvector xlin=x; // state for linearization
```

```
cvector v(2);
cvector dym=g(ym,p);
v[0]=dym[0]+ALPHA0*(ym[0]-y[0]);
v[1]=dym[1]+ALPHA1*(ym[1]-y[1]);
cvector w=invbeta(xlin)*v;
double re=w[0]-rvel;
ur=rpid.out(re);
double le=w[1]-lvel;
ul=lpid.out(le);
rov.rmotor=ur;
rov.lmotor=ul;
// Reference Model Simulation
ym=rk(ym,p,ST,g);
ODOMETRY *odo=new ODOMETRY;
odo->npr=npr;
odo->npl=npl;
odo->ur=ur;
odo->ul=ul;
odo->x=y[0];
odo->y=y[1];
odo->phi=x[2];
odolist.append(odo);
}
rov.off();
#ifdef REAL_TIME
close(timer);
#endif
if(!error)
{
```

```
cout << "Saving odometry data to disk...";
ofstream odofile("./twlckin.dat");
if (!odofile)
{
cout << "Can't open output file.\n\n";
return ;
}
ODOMETRY *odo;
int t=0;
while((odo=odolist.head()))
{
odofile << t++ << "\t";
odofile << odo->npr << "\t";
odofile << odo->npl << "\t";
odofile << odo->ur << "\t";
odofile << odo->ul << "\t";
odofile << odo->x << "\t";
odofile << odo->y << "\t";
odofile << odo->phi << "\n";
delete odo;
}
cout << "OK.\n";
}
fprintf(ctrllog,"Vou enviar UP pro principal\n");
sem_post(&sem_princ);
fprintf(ctrllog,"Terminando...\n");
fclose(ctrllog);
}
```

A01.2-Programa sugerido em ASM para controle dos motores das rodas

```
;***************************************************************************
;*************************** dipWM MotorDC 12 Volts **************************
;***************************************************************************
;
; Versão 1.1 em breve na versão 3
;
; Código Assembly para PIC16F84. Use dipASM ou MPLAB para assemblar.
;
; Default radix = hex. Números decimais são precedidos por"."(ex.".200")
; Calculado para usar o PIC de 4Mhz com oscilador de cristal e ; capacitores ceramicos de 1.6 pF.
;
; Valores máximos usados são para 12 volts e 120 watts (50%Torque/50%Velocidade).

list p=16F84                        ; lista de diretivas para este processador
#include <p16F84.inc>               ; inclui definições das variáveis específicas

; ******************** configuração inicial do WDT e cristal ************************
__CONFIG _CP_OFF & _WDT_OFF & _PWRTE_ON & _XT_OSC
;************************ DEFINIÇOES das VARIAVEIS *************************
;*************** As localizações de RAM do 16F84 iniciam em 0x0C *******************

hi_count    EQU 0x0c    ; registro de atraso (delay) superior
lo_count    EQU 0x0d    ; registro de atraso (delay) inferior
mainloop    EQU 0x0e    ; registro do pwm no loop
ld_count    EQU 0x0f    ; registro de atraso longo
ontime      EQU 0x10    ; registro de tempo em nivel 1
offtime     EQU 0x11    ; registro de tempo em nivel 0
but_tim     EQU 0x12    ; botão de temporização
```

```
;********************** DEFINIÇÕES DE CONSTANTES
*************************

dipwm       EQU 1       ; define bit 1 da PORTA B para a saída PWM(HEXFET)
codigo      EQU 7       ; define bit 7 da PORTA B para o botão de tempo

;************************************************************************
***

            ORG 0x000   ; vetor de reset do processador

;************************************************************************
***
;
; A execução do programa inicia aqui após o power up ou wake up do sleep
;
;************************************************************************
***
main
        clrf  PORTB             ; limpa a PORTA B
        bsf   STATUS,RP0        ; seleciona os registros do banco 1 (configurar)
        movlw b'10000000'       ; faz o bit 7 da PORTA B como entrada as outras são
saídas
        movwf TRISB             ; move conteúdo de W para o tristate da PORTA B
        bcf   STATUS,RP0        ; seleciona os registros do banco 0 (normal)
        movlw b'00001000'
        movwf INTCON            ; habilita o “wake up” na mudança da PORTA B
        movlw .100              ; pequeno atraso para prevenir partida acidental
        call  long_delay
        btfss PORTB,codigo      ; O “flag” de indicação de Ação do Supervisor foi
indicado?
        goto  power_down        ; se não - vá para “sleep”
pwr_on
        btfsc PORTB,codigo      ; o valor de Ação está sendo liberado?
        goto  pwr_on            ; (assegure que o modo não entre enquanto o valor de
Ação
```

```
                                          ; do “Flag” está sendo lido)

;******************************* modo de 30 watt
***********************************

;******************************************************************************
***

thirty_watt                    ; 25% do ciclo de trabalho é aprox. 30 watts
      movlw .19                ; mostra quanto tempo o valor deve ser passado
      movwf but_tim            ; desliga o motor (19 = aprox 2 Segundos)
      movlw .25                ; coloca o tempo de PWM "on" para 2.5 ms
      movwf ontime
      movlw .75                ; coloca o tempo de PWM "off" para 7.5 ms
      movwf offtime            ; \ (3.5 + 6.5 = 10 ms ou 100Hz a razão de PWM)
      call  runmotor           ; chama a subrotina do PWM

;******************************* modo de 60 watt
***********************************

;******************************************************************************
***   sixty_watt                     ; 60% duty cycle is aprox. 60  watts
      movlw .19                ; ídem
      movwf but_tim
      movlw .60
      movwf ontime
      movlw .40
      movwf offtime
      call  runmotor

;******************************* modo de 90 watt
**********************************

;******************************************************************************
***

seventy_watt                   ; 75% duty cycle is aprox. 75 watts
      movlw .19                ; ídem
      movwf but_tim
      movlw .75
      movwf ontime
      movlw .25                ; short off time to allow normal button scan
```

```
        movwf offtime
        call  runmotor
        goto  seven_watt
;**************************** modo de 120 watt
*******************************
;************************************************************************
***
hundry_twenty_watt              ; 100% duty cycle is full 120 watts
        movlw .19               ; ídem
        movwf but_tim
        movlw .100
        movwf ontime
        movlw .1                ; short off time to allow normal button scan
        movwf offtime
        call  runmotor
        goto  seven_watt

;****************************** Subrotinas
**********************************
;************************************************************************
***
; Subrotinas do dipWM e da varredura do valor do código com o valor da ação
; Ao ler rápido o programa cairá para o próximo nível de potência
; Ao ler demorado (2seg) vai para Power Down (sleep).
;************************************************************************
***
runmotor
        movlw  .7               ; coloca a razão de varredura do botão em aprox.14Hz
        movwf  mainloop         ; (7 x 10ms = 70ms)
run_d2
        bsf    PORTB,dipwm      ; liga o motor
        movf   ontime, 0        ; move o tempo de ligado( ontime ) para o registro W
        call   v_delay          ; chama a rotina da variavel de atraso
  bcf    PORTB,dipwm            ; desliga o motor
  movf   offtime, 0          ; move o tempo de desligado para o registro W
  call   v_delay             ; chama a rotina da variavel de atraso
  decfsz mainloop            ; decrementa o registro de PWM mainloop, se zero
```

```
   goto   run_d2                 ; então lê o botão senão mantêm em loop

   btfsc  PORTB,button                 ; lê o botão e se estiver sendo pressionado
   goto   timer_1                ; decrementa a variável em timer_1

         movlw  .19
         subwf  but_tim, 0             ; compara o valor no registro but_tim com 19
         btfsc  STATUS, Z              ; testa se o registro but_tim  = 19 ?

         goto runmotor                 ; se sim - o botão não foi tocado - continue PWM
         return                        ; se não - o botão foi pressionado e solto
                                 ; vá para o próximo nivel de potência
timer_1                                ; testa por quanto tempo o botão foi pressionado
   decfsz but_tim                ; decrementa but_tim, e testa se é = 0 ?
   goto runmotor                 ; se não - continue PWM
         goto power_down               ; se sim - o botão foi seguro por 2 segundos
                                 ; então abaixe a potência

;**********************************************************************
***
power_down                             ; liga o motor e dorme (modo de baixa potência)
         bcf   PORTB, pwm
         movlw .250                    ; atrasa 1 segundo para permitir que o botão seja solto
antes
         call  long_delay              ; colocando o processador para dormir
         movlw .250
         call  long_delay
         movlw .250
         call  long_delay
         movlw .250
         call  long_delay
         clrf  PORTB
         sleep                         ; acorda (botão pressionado) executará a prox. instrução
         goto  main
```

;****************************** subrotinas de atraso ******************************

;***

```
long_delay                      ; entre aquí com os ms desejados em W (max 255ms)
      movwf  ld_count           ; (1ms de resolução)
ld_loop
      call   ms_delay           ; chama atraso de 1ms
      decfsz ld_count           ; decrementa ld_count até encontrar o zero
      goto   ld_loop
      return

ms_delay                        ; entra aquí para atraso exato de 1 ms (.001 second)
   movlw .10              ; (1) carrega 10 no registro W
v_delay                   ;entra aquí para atraso com o desejado ms*10 em W (Max
                                ;25.5ms)
   movwf  hi_count              ; (1) (i.e. para atraso de 20 ms W = .200)(.1ms de
resolução)
loop1
   movlw  .31             ; (1)
   movwf  lo_count              ; (1)
loop2
   decfsz lo_count        ; (30x3)+2 = 92 ciclos (4MHz = 1 uS por ciclo de instrução)
   goto   loop2           ; /
   nop                          ; (1)
   decfsz hi_count        ; (1)
   goto   loop1           ; (2)
   return
   end
```

A02.Rotina do Matlab para a determinação do CG

```
% This program is for CG determination, specifically the
% Wheelchair's CG - Marcio Coelho & Carlos A. Rennó 04/02/02
```

```
clear

clear functions

clc

% Components parameters = [ xlenght ylenght zlenght mass ]

CG_roda_dianteira = [300 -20 0 280];

CG_roda_traseira_esquerda = [0 0 -210 895];

CG_roda_traseira_direita = [0 0 210 895];

CG_bateria_esquerda = [5 50 -32 2580];

CG_bateria_direita = [5 50 32 2580];

CG_redutor_esquerdo = [0 0 -110 330];

CG_redutor_direito = [0 0 110 330];

CG_motor_esquerdo = [-80 90 -110 1565];

CG_motor_direito = [-80 90 110 1565];

CG_plataforma = [160 0 0 1790];

CG_eletronica = [300 0 0 0];

% General matrix of the weelchair's parameters

M = [CG_roda_dianteira

     CG_roda_traseira_esquerda

     CG_roda_traseira_direita

     CG_bateria_esquerda

     CG_bateria_direita

     CG_redutor_esquerdo

     CG_redutor_direito

     CG_motor_esquerdo
```

```
    CG_motor_direito
    CG_plataforma
    CG_eletronica]

% CG Coordinates accordling to the x,y and z axes

xcg = sum( M(:,1).*M(:,4) )/sum(M(:,4))
ycg = sum( M(:,2).*M(:,4) )/sum(M(:,4))
zcg = sum( M(:,3).*M(:,4) )/sum(M(:,4))

% CG distances vector

di = [xcg ycg zcg]'

% CG angle between the axis

ANGXY = atan(ycg/xcg)% Angle between x and y axis
XY_Deg= ANGXY*180/pi% Radians to Degree conversion

ANGZY = atan(zcg/ycg);% Angle between z and y axis

ANGZX = atan(zcg/xcg);% Angle between z and x axis

% Plotting the 3-D surface of the wheelchair

[cx]=(M(:,1));
[cy]=(M(:,2));
[cz]=(M(:,3));
plot3(0,0,0,'bo',xcg,zcg,ycg,'ko',cx,cz,cy,'ro')
```

```
grid on
axis square

X=[0,xcg];
Y=[0,zcg];%Plot compatibilization inversion
Z=[0,ycg];%Iden
% Plotting the CG axis
line(X,Y,Z,'Color','k','LineWidth',2)
% Plotting the Wheels axis
XW=[0,0];
YW=[0,0];
ZW=[-210,210];
line(XW,ZW,YW,'Color','r','LineWidth',2)
```

A03.Rotina do Matlab para o modelamento matemático

A03.1.Programa Principal

```
% TsPg30 - Dynamic Simulation of Wheelchair's static Math model
% ================================================================
% Carlos A. Rönnow, 29-04-2001
% (C)opyright 2001
% Revisão: 3.1  - Data: 21/04/2002 - Hora: 18:37:00

echo on
deg2rad = pi/180;

% HOW INITIAL STATES FOR WheelChair's SIMULATION WERE OBTAINED
% =================================================================
```

```
% Below is the code that was used to define the initial states Pm
% consisting of various combinations of Angle, Velocity, and
% Force values, in addition to a set of steady state conditions.

Ang = [-20:22:200]*deg2rad;% Used # in Wheelchair?
Vel = [-90:36:90]*deg2rad;
For = -30:6:30;            % Next routine TsPg3a forces it to zero
An2 = [-20:10:200]*deg2rad;
Pm = [combvec(Ang,Vel,For) [An2; zeros(2,length(An2))]];

pause % Strike any key to see the WheelChair's simulation...

TsPg3a

% TsPg3a GENERATES DATABASE AND TsPg30 SAVES AS TsDd4a OF THE
INITIAL
% AND TARGETS STATES FOR THE SPECIFIC TsPg40 MODELING NEURAL NETWORK
%
===================================================================

save TsDd4a timestep Q Pm Tm p_time p_states init_state For

echo off
disp('End of TsPg30')
```

A03.2.Programa Simulador

```
% TsPg3a - Simulation Routine of Wheelchair's static math model
% ===================================================================
% Carlos A. Rönnow, 29-04-2001
```

```
% (C)opyright 2001

% Revisão: 3.1  - Data: 21/04/2002 - Hora: 18:37:00


clf;

figure(gcf)


echo on


% TESTING THE STATiC MATH MODEL TSMMLC10 TO SEE ITS BEHAVIOR

%
===================================================================

% To test the WheelChair's mathematics model, we will first
"measure"

% the WheelChair's response to these initial conditions (Velocity =
0,

% Force = 0)...


Ang = 5 * deg2rad;

Vel = 0*deg2rad;

For = 0;

init_state = [Ang; Vel];


% Here we "measure" and plot the WheelChair's response for 4
seconds.


p_time,p_states] = ode23('TsMmLP10',[0 4],[init_state; For]);

p_states = p_states';


pause % Strike any key to see the WheelChair's response...
```

```
% WheelChair math model RESPONSE

%
====================================================================

% Here we plot the WheelChair's Angle and Velocity responses.


figure(1)

title('Simulated WheelChair''s Response');

plot(p_time,p_states(1,:)/deg2rad,'b',p_time,p_states(2,:)/deg2rad,'
r')

grid on

xlabel('Time (sec)');

ylabel('Angle (deg) & Velocity (deg/sec)');

legend('Angulo','Velocidade');


% HOW TARGET STATE CHANGES ARE OBTAINED FOR FUTURE USE IN MODELING
ANN

%
====================================================================

% The target changes Tm were found by simulating the WheelChair's
math

% model TsMmLc10 for one time step for each initial state vector in
Pm.


timestep = 0.05;

Q = length(Pm);

Tm = zeros(2,Q);

for i=1:Q

   [ode_time,ode_state] = ode23('TsMmLp10',[0 timestep],Pm(:,i));

   Tm(:,i) = ode_state(length(ode_state),1:2)' - Pm(1:2,i);

end
```

```
for i=1:15
   for j=1:3
      figure(2)% Changes for best
      plot(ode_time,ode_state(i,j)/deg2rad,'ko-')
      grid on
      iu = rem(i,10);
      id = (i-iu)./10;
      Axis = axis
      AINT = [id+48,iu+48,j+48];
      STRG = char(AINT);
      text(Axis(2),Axis(4),STRG,'FontSize',14)
      xlabel('Time (sec)');
     ylabel('Force(Mass*Velocity/sec)');
     pause % Type any key to continue
   end
end

grid off
disp('End of TsPg3a')
```

A03.3.Função do Modelamento Matemático para Newton Simples

```
function dy=TsMmLc10(t,y)
%  TSMMLC10 Differential equation system for equilibrist WheelChair.
%  TsMmLc10(T,Y) - Equilibrist WheelChair Function
%    T - Time.
%    Y - Current state of equilibrist WheelChair.
%  Returns derivatives of the WheelChair state.
%
```

```
%  The state vector Y has three values:
%    Y(1) - WheelChair's angle from -2 pi to 2 pi radians.
%    Y(2) - WheelChair's angular velocity in radians/second.
%    Y(3) - Force being applied to the WheelChair.
%
%  NOTES: Angle is 0 radians when the WheelChair points up.
%         Force stays constant, its derivative is always 0.
%
%  See also TsPg60, PLINEAR.
% Carlos A. Rönnow, 29-04-2001
% Copyright (c) 2001
% Revisão: 1.12  - Data: 29/04/2001 - Hora: 15:37:22 $

if nargin < 2, error('Not enough input vectors.'); end

% STATE
Ang = y(1);
Vel = y(2);
For = y(3);

% CALCULATE DERIVATIVES
dAng = Vel;
dVel = 9.81*sin(Ang) - 2*Vel + For;
dFor = zeros(size(For));

% RETURN DERIVATIVES
dy = [dAng; dVel; dFor];
```

A03.4.Função do Modelamento Matemático para Lagrangeano

```
% SUBROTINAS PARA EFETUAR A DERIVAÇÃO DAS EQUAÇÕES DE LAGRANGE

% /////////// Inicio da Subrotina ///////////////////////////


clear

global H Q


% DETERMINAÇÃO DA ENERGIA CINÉTICA


n=4;

T1 = '1/2*m1*x5^2';

T2aux = 'x5^2+s^2*x6^2+2*s*x6*x5*cos(x2)';

T2 = symmul('1/2*m2/l',int(T2aux,'s',0'l'));

T3 = 0;


for j = 3:n,

   w = ['x' num2str(j) '*(a' num2str(j-2) '*(1/2*exp(b' num2str(j-2)

      '*s/l)-1/2/exp(b'num2str(j-2) '*s/l+sin(b'num2str(j-2)
'*s/l))'];

   wp = ['x' num2str(j+n) '*(a' num2str(j-2) '*(1/2*exp(b'
num2str(j-2)

      '*s/l)-1/2/exp(b'num2str(j-2) '*s/l+sin(b'num2str(j-2)
'*s/l))'];

   T31aux = symmul('2*(s*x6+x5+cos(x2))',wp);

   T31 = int(T31aux,'s',0,'l');

   T32aux=sympow(wp,2);

   T32 = int(T32aux,'s',0,'l');

   T33aux = '-2*x1*x6*sin(x2)';

   T33=symmul(T33aux,int(w,'s',0,'l'));
```

```
   T3 = symop(T3,'+','1/2*m2*l','*',symop(T31,'+',T32,'+',T33));
end

T = symop(T1,'+',T2,'+',T3);
T = simplify(T);
clear T1 T2 T2aux T3 T31 T31aux T32 T32aux T33 T33aux

% DETERMINAÇÃO DA ENERGIA POTENCIAL

V1 = 0;

for j = 3:n,
   w = ['x' num2str(j) '*(a' num2str(j-2) '*(1/2*exp(b' num2str(j-2)
      '*s/l)-1/2/exp(b'num2str(j-2) '*s/l+sin(b'num2str(j-2)
'*s/l))'];
   V11aux = sympow(diff(w,'s',2),2);
   V11 = symmu('El',int(V11aux,'s',0,'l');
   V12aux =(wp,2);
   V12aux = '-m2*g/l*sin(x2)';
   V12 = symmul(V12aux,int(w,'s',0,'l'));
   V1 = symadd(V1,symadd(V11,V12));
end

V2 = '1/2*m2*g/l*cos(x2)';
V = symadd(V11,V12);
V = symplify(V);
clear V1 V2 V11 V11aux V12 V12aux

% CÁLCULO DO LAGRANGEANO
```

```
L = symsub(T,V);
clear T V

%/////////// H.q = Q //////////////

H = [];
Q = [];

for k = 1:n,
   xk = ['x',num2str(k)];
   xkn = ['x',num2str(k+n)];
   dLdxp = dff(L,xkn);

   for i = 1:n*2
      xi = ['x',num2str(i)];
      H = [H,'',diff(dLdxp,xi)];
   end
   Q = [Q,'',diff(L,xk)];
end

H = ['[',H,']'];
Q = ['[',Q,']'];
clear L dLdxp i j k n w wp xi xk xkn

% /////////// Fim da Subrotina LAGRANGE /////////////////////////
```

A03.5.Função do Modelamento Matemático para Newton/Euler/Bryant

```
function dy=TsEqDiNe(t,y)
% FUNÇÃO TSEQDINE DE GERAÇÃO DAS EQUAÇÕES DIFERENCIAIS DA DINÂMICA
% DA CADEIRA EQUILIBRISTA PELO MÉTODO DE NEWTON/EULER
% TSEQDINE(T,Y)
%    T - Tempo.
%    Y - Estado Atual da cadeira.
% Retorna as derivadas do estado da cadeira.
%
% O vetor de estado Y tem nove valores para os deslocamento e suas
% derivadas e nove valores para as rotações e suas derivadas:
%    Y(1) - O deslocamento no eixo x em relação ao ref. Inercial.
%    Y(2) - A velocidade linear
%    Y(3) - Força segundo X sendo aplicado à cadeira.
%    Y(4) - O deslocamento no eixo y em relação ao ref. Inercial.
%    Y(5) - A velocidade linear
%    Y(6) - Força segundo Y sendo aplicado à cadeira.
%    Y(7) - O deslocamento no eixo z em relação ao ref. Inercial.
%    Y(8) - A velocidade linear
%    Y(9) - Força segundo Z sendo aplicado à cadeira.
%    Y(10) - O angulo Theta da cadeira de 0 pi to  pi radians.
%    Y(11) - A velocidade angular 'theta ponto' em radianos/segundo.
%    Y(12) - O Torque Theta sendo aplicado à cadeira.
%    Y(13) - O angulo Phi da cadeira de - pi/2 to pi/2 radians.
%    Y(14) - A velocidade angular 'phi ponto' em radianos/segundo.
%    Y(15) - O Torque Phi sendo aplicado à cadeira.
%    Y(16) - O angulo Psi da cadeira de - pi/3 to pi/3 radians.
%    Y(17) - A velocidade angular 'Psi ponto' em radianos/segundo.
%    Y(18) - O Torque Psi sendo aplicado à cadeira.
%
```

```
% OBS: Na prática, se a cadeira está apoiada, o ângulo theta é 0
rad.
%      sua velocidade é 0 rad/seg e a força permanece constante =
Mg.
%
% (C)opyright Carlos Antonio Rennó
% $Revisão: 1.0 $  $Data: 17/04/2002 17:35 $

if nargin < 2, error('Vetores de entrada insuficientes!'); end

% Condições iniciais ( com diferenciação para as variáveis no tempo
)

g = 9.81; % Gravidade
M = 2;
be=0.1;bd=0.1;bs=0.1; % Fricções diversas
r = 1; delta = 1;
Ixx = 1; Ixy = 1; Ixz = 1; Iyy =1; Iyz = 1; Izz = 1; % Inércias

% ESTADOS - entradas - detalhar
% LINEARES

xG  = y(1);%Coloca aqui o deslocamento atual segundo X
VxG = y(2);%Velocidade linear atual de X
FxG = y(3)
yG  = y(4);%Coloca aqui o deslocamento atual segundo Y
VyG = y(5);%Velocidade linear atual de Y
FyG = y(6)
zG  = y(7);%Coloca aqui o deslocamento atual segundo Z
VzG = y(8);%Velocidade linear atual de Z
```

```
FzG = y(9)

%ANGULARES

theta  = y(10);%Coloca aqui o angulo atual theta de rotação em Z
Vanthe = y(11);%Velocidade angular atual de theta
Torthe = y(12)
phi    = y(13);%Coloca aqui o angulo atual phi de rotação em Y
Vanphi = y(14);%Velocidades angular atual de phi
Torphi = y(15)
psi    = y(16);%Coloca aqui o angulo atual psi de rotação em X
Vanpsi = y(17);%Velocidade angular atual de psi
Torpsi = y(18)

% CALCULO DAS ACELERAÇÕES EM FUNÇÃO DAS FORÇAS E TORQUES APLICADOS
%
*******************************************************************
**
% CUIDADO -> CONFERIR SINAIS - e + devido às mudanças de lado na
equação

dxG = VxG;
%Força atual exercida em X (Transformada Inversa)
dVxG = - ( yG .* dVanpsi .* sin(psi) .* sin(theta)
+ zG .* dAngphi .* dAngthe .* sin(psi) .* sin(phi) .* cos(theta)
- xG .* cos(phi) .* cos(theta) .* (dAngthe).^2
- xG .* cos(phi) .* cos(theta) .* (dAngphi).^2
+ yG .* dVanthe .* cos(psi) .* sin(theta)
+ yG .* dVanpsi .* cos(psi) .* sin(theta)
- zG .* dVanphi .* sin(psi) .* sin(theta)
```

```
- zG .* dVanpsi .* sin(psi) .* sin(theta)

+ xG .* dAngthe .* dAngpsi .* cos(psi) .* sin(phi) .* cos(theta)

+ xG .* dVanthe .* dVanpsi .* sin(psi) .* sin(theta)

+ xG .* dAngphi .* dAngpsi .* sin(psi) .* sin(phi) .* cos(theta)

- xG .* dAngphi .* dAngpsi .* cos(psi) .* sin(theta)

- yG .* dVanthe .* sin(psi) .* sin(phi) .* cos(theta)

+ yG .* dAngthe .* dAngphi .* cos(psi) .* sin(phi) .* cos(theta)

+ yG .* dAngthe .* dAngphi .* sin(psi) .* sin(theta)

+ yG .* dAngpsi .* cos(phi) .* cos(theta) .* dAngphi

- yG .* dVanpsi .* sin(psi) .* sin(phi) .* cos(theta)

- zG .* dAngphi .* dAngthe .* cos(psi) .* sin(theta)

+ zG .* dAngpsi .* cos(phi) .* cos(theta) .* dAngthe

- zG .* dVanpsi .* cos(psi) .* sin(phi) .* cos(theta)

- xG .* dVanthe .* cos(psi) .* sin(theta)

- xG .* dVanphi .* sin(psi) .* sin(theta)

- yG .* cos(phi) .* cos(theta) .* dVanthe

+ zG .* cos(phi) .* cos(theta) .* dVanphi

+ zG .* dVanpsi .* cos(psi) .* sin(theta)

+ xG .* dVanthe .* sin(psi) .* sin(phi) .* cos(theta)

- xG .* dVanphi .* cos(psi) .* sin(phi) .* cos(theta)

+ yG .* dVanpsi .* cos(psi) .* sin(phi) .* cos(theta)

- zG .* dVanphi .* cos(psi) .* sin(phi) .* cos(theta)

- zG .* dVanpsi .* sin(psi) .* sin(phi) .* cos(theta))+ FxG/M;


%Calculo dos Torques em função da Força atual exercida em X
(Transformada %Direta)

%FxG= + ( Te .* cos(phi) .* cos(theta).^2 - Te .* sin(theta) .*
sin(psi) .* %sin(phi) .* cos(theta)

%+ Te .* cos(psi) - Te .* cos(psi) .* cos(theta).^2 + Td .* cos(phi)
.* %cos(theta).^2
```

```
%- Td .* sin(theta) .* sin(psi) .* sin(phi) .* cos(theta)

%+ Td .* cos(psi) - Td .* cos(psi) .* cos(theta).^2 )/r

dFxG = zeros(size(FxG));

%*******************************************************************
*******

dyG = VyG;

%Força atual exercida em Y(Transformada Inversa)

dVyG = - ( - xG .* cos(phi) .* sin(theta) .* (dAngthe).^2
- xG .* cos(phi) .* sin(theta) .* (dAngphi).^2
- yG .* (dAngthe).^2 .* cos(psi) .* cos(theta)
- yG .* (dAngpsi).^2 .* cos(psi) .* cos(theta)
+ zG .* (dAngphi).^2 .* sin(psi) .* cos(theta)
+ zG .* (dAngpsi).^2 .* sin(psi) .* cos(theta)
+ xG .* dAngthe .* dAngpsi .* cos(psi) .* sin(phi) .* sin(theta)
- xG .* dAngthe .* dAngpsi .* sin(psi) .* cos(theta)
+ xG .* dAngphi .* dAngpsi .* sin(psi) .* sin(phi) .* sin(theta)
+ xG .* dAngphi .* dAngpsi .* cos(psi) .* cos(theta)
- yG .* (dAngthe).^2 .*  sin(psi) .* sin(phi) .* sin(theta)
+ yG .* dAngthe .* dAngphi .* cos(psi) .* sin(phi) .* sin(theta)
- yG .* dAngthe .* dAngphi .* sin(psi) .* cos(theta)
+ yG .* dAngpsi .* cos(phi) .* sin(theta) .* dAngphi
- yG .* (dAngpsi).^2 .* sin(psi) .* sin(phi) .* sin(theta)
+ zG .* dAngphi .* dAngthe .* sin(psi) .* sin(phi) .* sin(theta)
+ zG .* dAngphi .* dAngthe .* cos(psi) .* cos(theta)
```

```
- zG .* (dAngphi).^2 .* cos(psi) .* sin(phi) .* sin(theta)

+ zG .* dAngpsi .* cos(phi)) .* sin(theta) .* dAngthe

- zG .* (dAngpsi).^2 .* cos(psi) .* sin(phi) .* sin(theta)

+ yG .* dVanpsi .* cos(psi) .* sin(phi) .* sin(theta)

+ xG .* dVanphi .* sin(psi) .* cos(theta)

- yG .* cos(phi) .* sin(theta) .* dVanthe

- yG .* dVanpsi .* sin(psi) .* cos(theta)

+ zG .* cos(phi)) .* sin(theta)) .* dVanphi

- zG .* dVanpsi .* cos(psi) .* cos(theta)

+ xG .* dVanthe .* sin(psi) .* sin(phi) .* sin(theta)

- xG .* dVanphi .* cos(psi) .* sin(phi) .* sin(theta)

- zG .* dVanpsi .* sin(psi) .* sin(phi) .* sin(theta)

+ xG .* dVanthe .* cos(psi) .* cos(theta))+ FyG/M;


%Calculo dos Torques em função da Força atual exercida em
Y(Transformada %Direta)

%FyG= + (cos(phi) .* sin(theta) .* cos(theta) .* Te + cos(phi) .*
sin(theta) .* %cos(theta) .* Td

%- sin(psi) .* sin(phi) .* Te + sin(psi) .* sin(phi) .* Te .*
cos(theta).^2

%- sin(psi) .* sin(phi) .* Td + sin(psi) .*  sin(phi) .*  Td .*
cos(theta).^2

%- sin(theta) .* cos(psi) .* cos(theta) .* Te

%- sin(theta) .* cos(psi) .* cos(theta) .* Td - M .* g .* r + R .*
r)/r


dFyG = zeros(size(FyG));


%**********************************************************************
***********
```

```
dzG = VzG;

%Força atual exercida em Z(Transformada Inversa)

dVzG = - ( xG .* sin(psi) .* cos(phi) .* dVanthe
- xG .* cos(psi) .* cos(phi) .* dVanphi
+ yG .* sin(phi) .* dVanthe
+ yG .* cos(psi) .* cos(phi) .* dVanpsi
- zG .* sin(phi) .* dVanphi
- zG .* sin(psi) .* cos(phi) .* dVanpsi
+ xG .* sin(phi) .* (dAngthe).^2
+ xG .* dAngthe .* cos(psi) .* cos(phi) .* dAngpsi
+ xG .* sin(phi) .* (dAngphi).^2
+ xG .* dAngphi .* sin(psi) .* cos(phi) .* dAngpsi
- yG .* sin(psi) .* cos(phi) .* (dAngthe).^2
+ yG .* dAngthe .* cos(psi) .* cos(phi) .* dAngphi
- yG .* dAngpsi .* sin(phi) .* dAngphi
- yG .* sin(psi) .* cos(phi) .* (dAngpsi).^2
+ zG .* dAngphi .* sin(psi) .* cos(phi) .* dAngthe
- zG .* cos(psi) .* cos(phi) .* (dAngphi).^2
- zG .* dAngpsi .* sin(phi) .* dAngthe
- zG .* cos(psi) .* cos(phi) .* (dAngpsi).^2) + FzG/M;

%Força atual exercida em Z(Transformada Direta)
%FzG= - (Te .* sin(phi) .* cos(theta) + Te .* sin(psi) .* cos(phi)
.* sin(theta)
%+ Td .* sin(phi) .* cos(theta) + Td .* sin(psi) .* cos(phi) .*
sin(theta))/r;

dFzG = zeros(size(FzG));
```

```
%**********************************************************************
*******

dAngthe = Vanthe;

%Torque devido a rotação theta(Transformada Inversa)

dVanthe  = ( 1/( Ixz .* M)) .* (+ Ixx .* dVanpsi - Ixy .* dVanphi
- dAngphi .* Ixz .* dAngpsi
- Iyz .* (dAngphi).^2 + dAngphi .* Izz .* dAngthe
+ dAngthe .* Ixy .* dAngpsi
- dAngthe .* Iyy .* dAngphi + Iyz .* (dAngthe).^2)
+ ( dVel(X) .* cos(phi) .* cos(theta) .* xG
+ dVel(X) .* yG .* sin(psi) .* sin(phi) .* cos(theta)
- dVel(X) .* yG .* cos(psi) .* sin(theta)
+ dVel(X) .* zG .* cos(psi) .* sin(phi) .* cos(theta)
+ dVel(X) .* zG .* sin(psi) .* sin(theta)
+ dVel(Y) .* cos(phi) .* sin(theta) .* xG
+ dVel(Y) .* yG .* sin(psi) .* sin(phi) .* sin(theta)
+ dVel(Y) .* yG .* cos(psi) .* cos(theta)
+ dVel(Y) .* zG .* cos(psi) .* sin(phi) .* sin(theta)
- dVel(Y) .* zG .* sin(psi) .* cos(theta)
- dVel(Z) .* sin(phi) .* xG
+ dVel(Z) .* sin(psi) .* cos(phi) .* yG
+ dVel(Z) .* cos(psi) .* cos(phi) .* zG ) - Torthe/( Ixz .* M );

%Calculo dos Torques devido a rotação theta(Transformada Direta)
%Torthe= - (delta .* Te .* sin(theta))/r + (delta .* Td .*
sin(theta))/r;
```

```
dTorthe = zeros(size(Torthe));

%********************************************************************
***********

dAngphi = Vanphi;

%Torque devido a rotação phi(Transformada Inversa)

dVanphi = ( 1/(Iyy.*M)) .* (- Ixy .* dVanpsi
- Iyz .* dVanthe + dAngthe .* Ixx .* dAngpsi
- dAngthe .* Ixy .* dAngphi - Ixz .* (dAngthe).^2
+ Ixz .* (dAngpsi).^2  + dAngpsi .* Iyz .* dAngphi
- dAngpsi .* Izz .* dAngthe)
+ ( dVel(X) .* cos(phi) .* cos(theta) .* xG
+ dVel(X) .* yG .* sin(psi) .* sin(phi) .* cos(theta)
- dVel(X) .* yG .* cos(psi) .* sin(theta)
+ dVel(X) .* zG .* cos(psi) .* sin(phi) .* cos(theta)
+ dVel(X) .* zG .* sin(psi) .* sin(theta)
+ dVel(Y) .* cos(phi) .* sin(theta) .* xG
+ dVel(Y) .* yG .* sin(psi) .* sin(phi) .* sin(theta)
+ dVel(Y) .* yG .* cos(psi) .* cos(theta)
+ dVel(Y) .* zG .* cos(psi) .* sin(phi) .* sin(theta)
- dVel(Y) .* zG .* sin(psi) .* cos(theta)
- dVel(Z) .* sin(phi) .* xG
+ dVel(Z) .* sin(psi) .* cos(phi) .* yG
+ dVel(Z) .* cos(psi) .* cos(phi) .* zG ) + Torphi/(Iyy .* M);
```

```
%Calculo dos Torques devido a rotação phi(Transformada Direta)

%Torphi= - ( delta .* Te .* cos(theta))/r + ( delta .* Td .*
cos(theta)))/r;


dTorphi = zeros(size(Torphi));


%*********************************************************************
***********


dAngpsi = Vanpsi;


%Torque devido a rotação psi(Transformada Inversa)


dVanpsi  = (1/(Ixz .* M) .* (- Iyz .* dVanphi

+ Izz .* dVanthe - Ixy .* (dAngpsi).^2

+ dAngpsi .* Iyy .* dAngphi

- dAngpsi .* Iyz .* dAngthe

- dAngphi .* Ixx .* dAngpsi + Ixy .* (dAngphi).^2

+ dAngphi .* Ixz .* dAngthe)

+ ( dVel(X) .* cos(phi) .* cos(theta) .* xG

+ dVel(X) .* yG .* sin(psi) .* sin(phi) .* cos(theta)

- dVel(X) .* yG .* cos(psi) .* sin(theta)

+ dVel(X) .* zG .* cos(psi) .* sin(phi) .* cos(theta)

+ dVel(X) .* zG .* sin(psi) .* sin(theta)

+ dVel(Y) .* cos(phi) .* sin(theta) .* xG

+ dVel(Y) .* yG .* sin(psi) .* sin(phi) .* sin(theta)

+ dVel(Y) .* yG .* cos(psi) .* cos(theta)

+ dVel(Y) .* zG .* cos(psi) .* sin(phi) .* sin(theta)

- dVel(Y) .* zG .* sin(psi) .* cos(theta)

- dVel(Z) .* sin(phi) .* xG
```

```
+ dVel(Z) .* sin(psi) .* cos(phi) .* yG

+ dVel(Z) .* cos(psi) .* cos(phi) .* zG ) + Torpsi/(Ixz .* M);


%Calculo dos Torques devido a rotação psi(Transformada Direta)

%Torpsi= sin(theta).^2 .* Te + cos(theta).^2 .* Te + sin(theta).^2
.* Td + %cos(theta).^2 .* Td;


dTorpsi = zeros(size(Torpsi));


%*********************************************************************
***********

% RETORNO DAS DERIVADAS, Forças e Torques calculados - Geral -
Detalhar

%*********************************************************************
***********


dy = [dxG; dVxG; FxG; dyG; dVyG; FyG; dzG; dVzG; FzG;

   dAngthe; dVanthe; Torthe; dAngphi; dVanphi;

   dTorphi; dAngpsi; dVanpsi; dTorpsi];


% Final da função TsEqDiNE
```

A04.Rotina do Matlab para o modelamento por dados

A04.1.Programa Principal

```
% TsPg40 - WheelChair Nonlinear system identification.

%
===================================================================

% Carlos A. Rönnow, 29-04-2001

% (C)opyright 2001
```

```
% Revisão: 3.1  - Data: 21/04/2002 - Hora: 18:37:00

echo on
rand('seed',584146048);

% TsPg30/TsPg3a generates TsDd4a database of the initial (Pm) and
% targets (Tm) states (IF IT DOESN'T EXISTS) for the modeling phase
% of the MODEL NEURAL NETWORK
%
=====================================================================

if ~exist('Tm','var'),load TsDd4a,
elseif ~exist('TsDd4a.MAT'),TsPg30,end

% NEWFF  - Creates feed-forward networks.
% TRAIN - Trains a network.
% SIM   - Simulates networks.
%
% NONLINEAR SYSTEM IDENTIFICATION:
%
=====================================================================
% Using the above functions a feed-forward network is trained to
model
% a nonlinear system: the equilibrist WheelChair.

pause % Strike any key to continue...

% DEFINING THE MODEL PROBLEM
%
=====================================================================
```

```
% We would like to train a network to model a WheelChair system
% described by the function TsMmLc10. To do this we need that
examples
% of WheelChair initial states and associated target state changes.

% DESIGNING THE MODELING NETWORK
%
================================================================
% NEWFF creates weights and biases for a two-layer TANSIG/PURELIN
% network with 8 TANSIG neurons.

S1 = 8;
[S2,Q] = size(Tm);

TsModNet = newff(minmax(Pm),[S1 S2],{'tansig' 'purelin'},'trainlm');

pause % Strike any key to train the neuron model...

% TRAINING THE NETWORK
%
================================================================
% We will use TRAIN to train the model network so that a typical
error
% is 0.0037 radians (0.2 deg) for the Q 2-element target vectors.

TsModNet.trainParam.show = 10;        % Freq. of progress (in
epochs).
TsModNet.trainParam.epochs = 300;  % Maximum numb. of epochs to
train.
TsModNet.trainParam.goal = (0.0037^2);% Mean-squared error goal.
```

```
pause % Training will begin... please wait... It takes a few minutes

TsModNet = train(TsModNet,Pm,Tm);

hold off

%    ...and finally finishes.

save TsDd4b TsModNet

pause % Strike any key to test the neuron model...

TsPg4a

echo off
hold off
grid off
disp('End of TsPg40')
```

A04.2.Programa Carregador de Dados e Simulador

```
% TsPg4a - Simulating Routine for Neural Networking Model
%
===================================================================
==
% Carlos A. Rönnow, 29-04-2001
% (C)opyright 2001
% Revisão: 3.1  - Data: 21/04/2002 - Hora: 18:37:00

deg2rad = pi/180;
```

```
if ~exist('TsModNet','var'),load TsDd4b,end

% WheelChair RESPONSE
% ===================================================================
===
% Here we plot the WheelChair's Angle and Velocity responses.
% OPEN LOOP RESPONSE
% ===================================================================
===
% Here we simulate the network's open loop response.

time = 0:timestep:4;
state = init_state;% It comes from TsPg30 through TsDd4a
states = zeros(2,length(time));
states(:,1) = state;
for i=2:length(time)
  state = state + sim(TsModNet,[state;For]);
  states(:,i) = state;
  echo off
end
echo on

pause % Strike any key to see the first comparison plot.

% WheelChair VS. OPEN LOOP MODEL's RESPONSES
% ===================================================================
====
```

```
% The open loop response can be compared with the WheelChair.

figure(1)
plot(p_time,p_states(1,:)/deg2rad,'k',time,states(1,:)/deg2rad,'or',
p_time,p_states(2,:)/deg2rad,'k',time,states(2,:)/deg2rad,'ob')
grid on
xlabel('Time (sec)');
ylabel('Ang.(deg)& Vel.(deg/sec) = o N -');
legend('Ref.Ang.','Sim.Ang.','Ref.Veloc.','Sim.Veloc.',0);
title('WheelChair and Open Network Response');

% Note that the model has learned the basic behavior of the
% WheelChair well.  However, over time the slight errors in
% the model accumulate.
% Fortunately, the model does not need to be used in an open
% loop manner.

pause % Strike any key to calculate the model's closed loop
response...

% MODEL CLOSED LOOP RESPONSE
%
=====================================================================
======
% Here we simulate the network's closed loop response.
% Each time step the network calculates its estimate of the
% WheelChairs next state, and then measurements of the WheelChair
% state are used to correct this estimate.

state = init_state;
```

```
states(:,1) = state;

for i=2:length(time)

  new_state = state + sim(TsModNet,[state; For]);

  states(:,i) = new_state;

  [ode_time,ode_state] = ode23('TsMmLp10',[0 timestep],[state;
For]);

  state = ode_state(length(ode_state),1:2)';

  echo off

end

echo on


pause % Strike any key to see the second comparison plot.


% WheelChair VS. CLOSED LOOP MODEL RESPONSES

%
=====================================================================
=======

% The closed loop response can be compared with the WheelChair.


figure(2)

plot(p_time,p_states(1,:)/deg2rad,'k',time,states(1,:)/deg2rad,'or',
p_time,p_states(2,:)/deg2rad,'k',time,states(2,:)/deg2rad,'ob')

grid on

xlabel('Time (sec)');

ylabel('Angle (deg)& Velocity (deg/sec) W x N -');

title('WheelChair and Closed Network Response');

legend('Ref.Ang.','Sim.Ang.','Ref.Veloc.','Sim.Veloc.',0);


% As can be seen from the plot, the closed loop response of

% the network model is extremely accurate.
```

```
% See TsPg60 to see how to design a controller using the model.

disp('End of TsPg4a')
```

A5. Rotina do Matlab para o controle moderno

```
% SUBROTINAS PARA EFETUAR A LINEARIZAÇÃO, CONTROLE E SIMULAÇÃO

% /////////// Inicio da Subrotina //////////////////////////////

function xp=contr_1(t,x)% Equação sem Observador de estado

% Equação para controle proporcional(com A,B,K calculados
anteriormente)

global A B K

xp=(A-B*K)*x;

function xp=contr_2(t,x)% Equação com Observador de estado

% Equação para controle proporcional(com A,B,C,K e Ke calculados
anteriormente)

global A B C K Ke

xp=[A-B*K B*K; zeros(size(A)) A-Ke*C]*x;

%Linearização, controle e simulação das equações do movimento

global E EI m1 m2 I mu H Q xo n A B C K Ke

n=4;

I=4.17 e-13;

E=200e9;

EI=E*I;

l=0.5;% comprimento do pendulo
```

```
m1=2;% massa do carro
m2=0.1;% massa do pendulo
xo=[0;0;0;0;0;0;0;0];
p=20;

% Linearização
[A,B,C,D]=linmod('eqpend');
C=[1 0 0 0 0 0 0 0];

% Controle com o calculo das matrizes de ganho( K e Ke )
L=poly(A);
Ac=[-2-0.5i -2+0.5i -p*ones(1,2*n-2)];
Ace=Ac*47;
Lc=poly(Ac);
Lce=poly(Ace);
% Matriz de controlabilidade
Mi=[B A*B A^2*B A^3*B A^4*B A^5*B A^6*B A^7*B];
% Matriz de observabilidade
N=[C' A'*C' A'^2*C'A'^3*C'A'^4*C' A'^5*C' A'^6*C' A'^7*C'];
W=[L(8) L(7) L(6) L(5) L(4) L(3) L(2) 1;
   L(7) L(6) L(5) L(4) L(3) L(2) 1    0;
   L(6) L(5) L(4) L(3) L(2) 1    0    0;
   L(5) L(4) L(3) L(2) 1    0    0    0;
   L(4) L(3) L(2) 1    0    0    0    0;
   L(3) L(2) 1    0    0    0    0    0;
   L(2) 1    0    0    0    0    0    0;
   1    0    0    0    0    0    0    0];
T=Mi*W;
Qi=inv(W*N');
```

```
disp('Ganho Requerido =');

K=[Lc(9)-L(9) Lc(8)-L(8) Lc(7)-L(7) Lc(6)-L(6) Lc(5)-L(5) Lc(3)-L(3)
Lc(2)-L(2)]*inv(T);

Ke=Qi*[Lce(9)-L(9) Lce(8)-L(8) Lce(7)-L(7) Lce(6)-L(6) Lce(5)-L(5)
Lce(3)-L(3) Lce(2)-L(2)]';


% Simulação das equações

to=0;

tf=4;

xoo=[0;0.5;0;0;0;0;0;0; 0;0.5;0;0;0;0;0;0];

[t,x]=ode45('contr_1',[to tf],xoo(1:8));% Sem observador

[t1,x1]=ode45('contr_2',[to tf],xoo);% Com observador


%///////// Fim da Subrotina CONTROLE ///////////////////////////
```

A06. Rotina do Matlab para o controle Não-Convencional

A06.1. Programa Principal do Controle baseado em dados

```
% TSPG60 -  Feedback Control Linearization.

%
====================================================================

% Carlos A. Rönnow, 29-04-2001

% (C)opyright 2001

% Revisão: 3.1  - Data: 21/04/2002 - Hora: 18:37:00


echo off

clf;

figure(gcf)
```

```
rand('seed',24792483);

echo on


% NEWFF - Creates feed-forward networks below.

% TRAIN - Trains a network.

% SIM   - Simulates networks.

%

% NONLINEAR SYSTEM CONTROL IDENTIFICATION:

%
====================================================================

% Using the above functions a feed-forward network is trained

% to control a nonlinear system: the equilibrist WheelChair.


% The control network is trained by using a model network to

% backpropagate system errors to the controller.


pause % Strike any key to continue...


% DEFINING THE CONTROL PROBLEM

%
====================================================================

% We would like to train a network to control a WheelChair so that
the

% entire system behaves according to the linear reference system
PLINEAR.

% To do this we need the anterior neural model of the WheelChair and

% examples of initial states and associated target next states.

% The file TsDd6a contains initial/target states and TsDd6b the
weights

% and biases for the model network.
```

```
if ~exist('TsDd6a.MAT'),TsPg6a,
   elseif ~exist('TsDd6b.MAT'),

pause % Strike any key to see where these values came from...

% HOW NETWORK MODEL AND INITIAL STATES WERE OBTAINED
%
===================================================================
% The network TsModNet which models the WheelChair was found with
the
% application script TSPG40.
% The file TsDd4b contains TSMODNET the weights and biases for the
% model network, and initial/target states.

   if ~exist('TsDd4b.MAT'),TsPg40,
   else load TsDd4b,end

pause % Strike any key to design the control network...

% DESIGNING THE CONTROL NETWORK
%
===================================================================
% Now, the function NEWFF creates weights and biases for a two-layer
% TANSIG/PURELIN network with 8 TANSIG neurons and 1 output.

   S1 = 8;
   TesCoNet = newff(minmax(Pc),[S1 1],{'tansig' 'purelin'});

% Now, function NETWORK creates a transiction network TNET which is
a
```

```
% combination of control network TesCoNet and model network
TsModNet.

% Only the weights and biases for the model network will be
adjusted.


   numInputs = 2;

   numLayers = 4;

   tnet = network(numInputs,numLayers);

   tnet.biasConnect = [1 1 1 1]';

   tnet.inputConnect = [1 0 1 0; 1 0 0 0]';

   tnet.layerConnect = [0 0 0 0; 1 0 0 0; 0 1 0 0; 0 0 1 0];

   tnet.outputConnect = [0 0 0 1];

   tnet.targetConnect = [0 0 0 1];

   tnet.inputs{1}.range = minmax(Pc(1:2,:));

   tnet.inputs{2}.range = minmax(Pc(3,:));

   tnet.layers{1}.size = S1;

   tnet.layers{1}.transferFcn = 'tansig';

   tnet.layers{2}.size = 1;

   tnet.layers{2}.transferFcn = 'purelin';

   tnet.layers{3}.size = 8;

   tnet.layers{3}.transferFcn = 'tansig';

   tnet.layers{4}.size = 2;

   tnet.layers{4}.transferFcn = 'purelin';

   tnet.performFcn = 'mse';

   tnet.trainFcn = 'trainbfg';

   tnet.IW{1,1} = TesCoNet.IW{1,1}(:,1:2);

   tnet.inputWeights{1,1}.learn = 1;

   tnet.IW{1,2} = TesCoNet.IW{1,1}(:,3);

   tnet.inputWeights{1,2}.learn = 1;

   tnet.b{1} = TesCoNet.b{1};
```

```
   tnet.biases{1}.learn = 1;

   tnet.b{2} = TesCoNet.b{2};

   tnet.biases{2}.learn = 1;

   tnet.LW{2,1} = TesCoNet.LW{2,1};

   tnet.layerWeights{2,1}.learn = 1;

   tnet.IW{3,1} = TsModNet.IW{1,1}(:,1:2);

   tnet.inputWeights{3,1}.learn = 0;

   tnet.LW{3,2} = TsModNet.IW{1,1}(:,3);

   tnet.layerWeights{3,2}.learn = 0;

   tnet.b{3} = TsModNet.b{1};

   tnet.biases{3}.learn = 0;

   tnet.LW{4,3} = TsModNet.LW{2,1};

   tnet.layerWeights{4,3}.learn = 0;

   tnet.b{4} = TsModNet.b{2};

   tnet.biases{4}.learn = 0;


pause % Strike any key to train the neural controller...


% TRAINING THE CONTROL NETWORK

%
====================================================================
====

% We will use TRAIN to train the control network so that

% a typical error is 0.002 radians (0.11 deg) for the Q

% 2-element target vectors.


   tnet.trainParam.show = 5;    % Frequency of progress displays (in
epochs).

   tnet.trainParam.epochs = 600;      % Maximum number of epochs to
train.
```

```
   tnet.trainParam.goal = (0.002^2);  % Sum-squared error goal.

% Training begins...please wait...

   [tnet,tr] = train(tnet,{Pc(1:2,:);  Pc(3,:)},{Tc});

% ...and finally finishes.

% Now we take the trained weights and put them back in the control
network

% to determine the final inverse control network.

   TesCoNet.IW{1,1}(:,1:2) = tnet.IW{1,1};

   TesCoNet.IW{1,1}(:,3) = tnet.IW{1,2};

   TesCoNet.b{1} = tnet.b{1};

   TesCoNet.b{2} = tnet.b{2};

   TesCoNet.LW{2,1} = tnet.LW{2,1};

   save TsDd6b cW1 cb1 cW2 cb2

else load TsDd6a, load TsDd6b,

end

pause % Strike any key to test the neuron model...

% SIMULATING THE CONTROL NETWORK

%
=====================================================================
====

TsPg6b
```

```
disp('End of TsPg60')
```

A06.2.Programa Carregador de Dados

```
% TSPG6A - Linear Model Reference Generator
% ====================================================================
==
% Carlos A. Rönnow, 29-04-2001
% (C)opyright 2001
% Revisão: 3.1  - Data: 21/04/2002 - Hora: 18:37:00

deg2rad = pi/180;

% HOW NETWORK MODEL TsModNet AND IT's INITIAL STATES WERE OBTAINED
% ====================================================================
==
% The network TsModNet which models the WheelChair was found with
the
% application script TsPg40.
%
% HOW INITIAL STATES WERE OBTAINED FOR USE IN THE LINEAR REFERENCE
MODEL
% ====================================================================
==
% Below is the code that was used to define the new set initial
states Pc
% consisting of various combinations of angle, velocity, and demand
values
```

```
% in addition to a set of steady state conditions(vel = 0, demand =
angle).

angle = [-10:40:190]*deg2rad;
vel = [-90:36:90]*deg2rad;
demand = [-180:40:180]*deg2rad;
angle2 = [-10:10:190]*deg2rad;
Pc = [combvec(angle,vel,demand) [angle2; zeros(size(angle2));
angle2]];

pause % Strike any key to see how the next states were obtained...

% HOW TARGET NEXT STATES WERE OBTAINED FOR LINEAR REFERENCE MODEL
%
====================================================================
==
% The target states Tc were found by simulating the desired linear
system
% PLINEAR for one time step for each initial state vector in Pc.

timestep = 0.05;
Q = length(Pc);
Tc = zeros(2,Q);
for i=1:Q
   [ode_time,ode_state] = ode23('plinear',[0 timestep],Pc(:,i));
   Tc(:,i) = ode_state(length(ode_state),1:2)' - Pc(1:2,i);
end

save TsDd6a Q timestep Pc Tc

disp('End of TsPg6a')
```

A06.3.Programa Simulador

```
% TSPG6B - WheelChair's Intelligent Inverse Control Testing
% =================================================================
=
% Carlos A. Rönnow, 29-04-2001
% (C)opyright 2001
% Revisão: 3.1  - Data: 21/04/2002 - Hora: 18:37:00

deg2rad = pi/180;

if ~exist('TsModNet'),load TsDd4b,end
if ~exist('TesCoNet'),load TsDd6b,end
timestep = 0.05;
echo on

% TESTING THE CONTROLLER
% ======================
% Here are some initial conditions for which the controller
% will be tested:

ang0 = 10*deg2rad;
vel0 = 0;
init_state = [ang0; vel0];

% Here is the desired angle of the WheelChair:

demand = 90*deg2rad;
```

```
pause % Strike any key to calculate the desired response...

% DESIRED BEHAVIOR
% ================
% The function ODE23 can be used to calculate the desired
% response of the controlled WheelChair.  The controlled WheelChair
% is to behave like the linear system PLINEAR.

[p_time,p_states] = ode23('plinear',[0 4],[init_state; demand]);
p_states = p_states';

pause % Strike any key to see the desired response...

% DESIRED RESPONSE
% ================
% Here we plot the desired angle and velocity responses.

figure(1)
plot(p_time,p_states(1,:)/deg2rad,'k',p_time,p_time*0+demand/deg2rad
,'or')
grid on
hold on
plot(p_time,p_states(2,:)/deg2rad,'k',p_time,p_time*0,'ob')
grid on
xlabel('Time');
ylabel('Angle (deg)&Velocity (deg/sec)');
title('WheelChair''s Response');
legend('Des.Ang.','Sim.Ang.','Def.Veloc.','Sim.Veloc.',0);
```

```
% Note that the angle quickly moves to 90 degrees and to
% a velocity of 0.

pause % Strike any key to calculate the actual response...

% ACTUAL BEHAVIOR
% ===============
% Each time step the control network calculates the right
% force to make the WheelChair behave like the linear reference
% model.

time = 0:timestep:4;
state = init_state;
states = zeros(2,length(time));
states(:,1) = state;
for i=2:length(time)
  force = sim(TesCoNet,[state; demand]);
  [ode_time,ode_state] = ode23('TsMmLp10',[0 timestep],[state;
force]);
  state = ode_state(length(ode_state),1:2)';
  states(:,i) = state;
  echo off
end
echo on

pause % Strike any key to see the actual response...

% ACTUAL RESPONSE
```

```
%
====================================================================
=

% Here we plot the actual response of the controlled WheelChair

% and compare it to the desired linear system response.


figure(2)

plot(p_time,p_states(1,:)/deg2rad,'+r',time,states(1,:)/deg2rad,'k')

grid on

hold on

plot(p_time,p_time*0+demand/deg2rad,'or')

xlabel('Time (sec)');

ylabel('Angle (deg): D + A -');

title('WheelChair''s Desired and Actual Response');

plot(p_time,p_states(2,:)/deg2rad,'+r',time,states(2,:)/deg2rad,'k')

hold on

plot(p_time,p_time*0,'ob')

xlabel('Time (sec)');

ylabel('Velocity (deg/sec): D+ A -');

legend('Des.Ang.','Sim.Ang.','Def.Veloc.','Sim.Veloc.',0);


% The response is close to perfect. The WheelChair has been
controlled

% to match the linear reference model with a neural controller.


hold off

echo off

disp('End of TsPg6b')
```

A06.4.Função da Referência Linear

```
function dy=plinear(t,y)
%PLINEAR Differential equation system for desired linear pendulum.
%
%  PLINEAR(T,Y)
%    T - Time.
%    Y - Current state of inverted pendulum.
%  Returns derivatives of the pendulum state.
%
%  The state vector Y has three values:
%    Y(1) - Pendulum angle from -2 pi to 2 pi radians.
%    Y(2) - Pendulum angular velocity in radians/second.
%    Y(3) - Demand angle for the pendulum.
%
%  NOTES: Angle is 0 radians when the pendulum points up.
%         Demand stays constant, its derivative is always 0.
%
%  See also APPCS1, PMODEL.

% Mark Beale, 12-15-93
% Copyright (c) 1992-1998 by the MathWorks, Inc.
% $Revision: 1.6 $  $Date: 1997/12/30 19:38:45 $

if nargin < 2,error('Not enough input arguments.'), end

% STATE
angle  = y(1);
vel    = y(2);
demand = y(3);
```

```
% CALCULATE DERIVATIVES
dangle  = vel;
dvel    = -9*angle - 6*vel + 9*demand;
ddemand = 0;

% RETURN DERIVATIVES
dy = [dangle; dvel; ddemand];

%
==================================================================
=
```

A06.2.Controle baseado em conhecimento (Lógica Nebulosa ou Fuzzy)
A06.2.1.Fuzificação das variáveis de entrada em Lógica Nebulosa (ou Fuzzy)

```
function fuzzyfication

%Implements the fuzification of the input variables
% FUZZIFICATION - collect Wheelchair dynamic data
% data format:
% [y(k) y(k-1) y(k-2) y(k-3) y(k-4)
%  u(k-1) u(k-2) u(k-3) u(k-4) u(k-5) u(k-6)]
% Copyright (c) 2001 by Carlos A. Ronnow
% $Revision: 1.1 $

Chairdat; %Inputs Wheelchair data

group1 = [1 2 3 4];% y(k-1), y(k-2), y(k-3), y(k-4)
group2 = [5 6 7 8 9 10]; % u(k-1) through y(k-6)
% Copyright (c) 1994-98 by The MathWorks, Inc.
```

```
% $Revision: 1.2 $

anfis_n = length(group1)*length(group2);
index = zeros(anfis_n, 2);
trn_error = zeros(anfis_n, 1);
chk_error = zeros(anfis_n, 1);
% ======= Training options
mf_n = 2;
mf_type = 'gbellmf';
epoch_n = 1;
ss = 0.1;
ss_dec_rate = 0.5;
ss_inc_rate = 1.5;
% ====== Train ANFIS with different input variables
fprintf('\nTrain %d ANFIS models, each with 2 inputs selected from
10 candidates...\n\n',...
   anfis_n);
model = 1;
tic
for i=1:length(group1),
  for j=1:length(group2),
   in1 = deblank(input_name(group1(i), :));
   in2 = deblank(input_name(group2(j), :));
   fprintf('ANFIS model %d: %s %s\n', model, in1, in2);
   index(model, :) = [group1(i) group2(j)];
   trn_data = data(1:trn_data_n, [group1(i) group2(j)
size(data,2)]);
   chk_data = data(trn_data_n+1:size(data,1), [group1(i) group2(j)
size(data,2)]);
   in_fismat = genfis1(trn_data, mf_n, mf_type);
```

```
    [trn_out_fismat t_err step_size chk_out_fismat c_err] = ...
      anfis(trn_data, in_fismat, ...
      [epoch_n nan ss ss_dec_rate ss_inc_rate], ...
      [0 0 0 0], chk_data, 1);
    trn_error(model) = min(t_err);
    chk_error(model) = min(c_err);
    model = model+1;
  end
end
toc

% ====== Reordering according to training error
[a b] = sort(trn_error);
%b = flipud(b);     % List according to decreasing trn error
trn_error = trn_error(b);
chk_error = chk_error(b);
index = index(b, :);
% ====== Display training and checking errors
figTitle = 'ANFIS: Input Selection';
figH = findobj(0, 'name', figTitle);
if isempty(figH),
   figH = figure(...
      'Name', figTitle, ...
      'NumberTitle', 'off');
else
   set(0, 'currentfig', figH);
end

x = (1:anfis_n)';
```

```
figure;

subplot(211);

plot(x, trn_error, '-', x, chk_error, '-', ...

     x, trn_error, 'o', x, chk_error, '*');

tmp = x(:, ones(1, 3))';

X = tmp(:);

tmp = [zeros(anfis_n, 1) max(trn_error, chk_error) nan*ones(anfis_n,
1)]';

Y = tmp(:);

hold on; plot(X, Y, 'g'); hold off;

axis([1 anfis_n -inf inf]);

set(gca, 'xticklabel', []);


% ====== Add text of input variables

for k = 1:anfis_n,

   text(x(k), 0, ...

   [input_name(index(k,1), :) ' ' ...

    input_name(index(k,2), :)]);

end

h = findobj(gcf, 'type', 'text');

set(h, 'rot', 90, 'fontsize', 11, 'hori', 'right');

drawnow


% ====== Generate input_index for bjtrain.m

[a b] = min(trn_error);

input_index = index(b,:);

title('Training (Circles) and Test (Asterisks) Errors');

ylabel('RMSE');
```

```
data_n = length(y2);
output = y2;
input = [[0; y2(1:data_n-1)] ...
    [0; 0; y2(1:data_n-2)] ...
    [0; 0; 0; y2(1:data_n-3)] ...
    [0; 0; 0; 0; y2(1:data_n-4)] ...
    [0; u2(1:data_n-1)] ...
    [0; 0; u2(1:data_n-2)] ...
    [0; 0; 0; u2(1:data_n-3)] ...
    [0; 0; 0; 0; u2(1:data_n-4)] ...
    [0; 0; 0; 0; 0; u2(1:data_n-5)] ...
    [0; 0; 0; 0; 0; 0; u2(1:data_n-6)]];
data = [input output];
data(1:6, :) = [];
input_name = str2mat('y(k-1)', 'y(k-2)', 'y(k-3)', 'y(k-4)', ...
   'u(k-1)', 'u(k-2)', 'u(k-3)', 'u(k-4)', 'u(k-5)', 'u(k-6)');
%input_name = str2mat('y(k)', 'y(k-1)', 'y(k-2)', 'y(k-3)', ...
%  'u(k)', 'u(k-1)', 'u(k-2)', 'u(k-3)', 'u(k-4)', 'u(k-5)');

trn_data_n = 300;  % No. of training data pairs
index = 1:100;
subplot(2,1,1); plot(index, y2(index), '-', index, y2(index), 'o');
ylabel('y(k)');
subplot(2,1,2); plot(index, u2(index), '-', index, u2(index), 'o');
ylabel('u(k)');
```

A06.2.2.Programa do Controle em Lógica Nebulosa (ou Fuzzy)

```
function fuzzycontrol

%Implements the fuzzy control of the wheelchair

    stepH = h(2); set(stepH, 'tag', 'step');
    %===== put UI handles into current figure's user data
    tmp = [startH stopH pauseH contH stepH countH -1 -1 -1 -1];
    set(InvKineFigH, 'userdata', tmp, 'HandleVisibility', 'on');

%elseif strcmp(action, 'set_extra_gui'),    % extra UI's
    % ====== extra UI
    % ====== The upper UI controls (Specific to each animation)
    cb1 = [mfilename '(''show_trail'')'];
    cb2 = [mfilename '(''clear_trail'')'];
    cb3 = '';
    cb4 = [mfilename '(''target_pos'')'];

    string1 = 'Show Trails';
    string2 = 'Clear Trails';
    string3 = 'Using mouse to drag target oval inside yellow x
marked area';
    string4 = 'Ellipse|Mouse-Driven|';

    [upH, upPos] = uiarray(Pos(1,:), 1, 4, spacing, 1*spacing, ...
        str2mat('check', 'push', 'text', 'popup'), ...
        str2mat(cb1, cb2, cb3, cb4), ...
        str2mat(string1, string2, string3, string4));
    set(upH(3), 'HorizontalAlignment', 'right');
    showTrailH = upH(1); set(showTrailH, 'tag', 'show_trail');
```

```
    signalH = upH(4);
    % The following will be put back if we have more than one
    % desired trajectory
    thispos1=get(upH(2),'Position');
    thispos1(1)=thispos1(1)*.7;
    thispos1(3)=thispos1(3)*.8;
    set(upH(2),  'Position', thispos1);
    thispos=get(upH(3),'Position');
    thispos(3)=thispos(3)*2.4;
    thispos(1)=thispos(1)*.8;
    set(upH(3),  'Position', thispos);
%    delete(upH(3));
    delete(upH(4));
    % ====== Appending handles as the second row of userdata
    tmp = [signalH showTrailH -1 -1 -1 -1 -1 -1 -1 -1];
    set(InvKineFigH, 'userdata', [get(InvKineFigH, 'userdata');
tmp]);

    % ====== change labels of standard UI controls
%   tmp = get(InvKineFigH, 'userdata');
%   set(tmp(1, 1), 'visible', 'off');
%   set(tmp(1, 2:3), 'visible', 'on');

elseif strcmp(action, 'set_mouse_action'),
    % action when button is first pushed down
    action1 = [mfilename '(''mouse_action1'')'];
    % actions after the mouse is pushed down
    action2 = [mfilename '(''mouse_action2'')'];
    % action when button is released
```

```
    action3 = [mfilename '(''mouse_action3'')'];

    % temporary storage for the recall in the down_action
    set(gca,'UserData',action2);

    % set action when the mouse is pushed down
    down_action=[ ...
        'set(gcf,''WindowButtonMotionFcn'',get(gca,''UserData''));'
...
        action1];
    set(gcf,'WindowButtonDownFcn',down_action);

    % set action when the mouse is released
    up_action=[ ...
        'set(gcf,''WindowButtonMotionFcn'','' '');', action3];
    set(gcf,'WindowButtonUpFcn',up_action);
elseif strcmp(action, 'set_init_cond'),
    % This is called whenevern going from "stop" to "start"
    InvKineCount = 1;
    % set desired trajectory
    r1 = 5; r2 = 3;          % axes for reference ellipse
    center = 11*exp(j*pi/4);    % center for reference ellipse
    data_n = 50;
    t = linspace(0, 2*pi, data_n)-pi/2;
    desired_traj = r1*cos(t) + j*r2*sin(t) +  center;
    tmp = get(findobj(0, 'name', InvKineFigTitle), 'userdata');
    desired_trajH = tmp(3, 4);
    set(desired_trajH, 'xdata', real(desired_traj));
    set(desired_trajH, 'ydata', imag(desired_traj));
```

```
    % set actually trajectory
    actual_trajH = tmp(3, 5);
    data = desired_traj(1)*ones(2,1);
    set(actual_trajH, 'xdata', real(data));
    set(actual_trajH, 'ydata', imag(data));
    eval([mfilename, '(''clear_trail'')']);
elseif strcmp(action, 'set_anim_obj'),
    l1 = 10; l2 = 7;        % specifications for robot arms
    r1 = 5; r2 = 3;         % axes for reference ellipse
    % ====== arm 1
    init_pos = [0; l1] + j*[0; 0];
    arm1H = line(real(init_pos), imag(init_pos));
    set(arm1H, 'userdata', l1);
    set(arm1H, 'erasemode', 'xor', 'color', 'r', 'linewidth', 4);
    set(arm1H, 'clipping', 'off');
    % ====== arm 2
    init_pos = [l1; l1+l2] + j*[0; 0];
    arm2H = line(real(init_pos), imag(init_pos));
    set(arm2H, 'userdata', l2);
    set(arm2H, 'erasemode', 'xor', 'color', 'c', 'linewidth', 4);
    set(arm2H, 'clipping', 'off');
    % ====== small circle showing actual trajectory
    dotH = line(0, 0);
    set(dotH, 'Marker', 'o', 'color', 'g', 'erasemode', 'xor');
    set(dotH, 'linewidth', 2);
    % ====== desired trajectory
    center = 11*exp(j*pi/4);    % center for reference ellipse
    data_n = 50;
    t = linspace(0, 2*pi, data_n)-pi/2;
```

```
    desired_traj = r1*cos(t) + j*r2*sin(t) +  center;
    desired_trajH = line(real(desired_traj), imag(desired_traj));
    set(desired_trajH, 'erasemode', 'xor');
    set(desired_trajH, 'linewidth', 2, 'userdata', desired_traj);
    set(desired_trajH, 'clipping', 'off');
    % ====== actual trajectory
    actual_trajH = line([1 1]*real(desired_traj(1)), ...
        [1 1]*imag(desired_traj(1)));
    set(actual_trajH, 'erase', 'none', 'color', 'w');
    set(actual_trajH, 'clipping', 'off');
    set(actual_trajH, 'linewidth', 2, 'linestyle', '--');
    % ====== setting axis
    axis([0 l1+l2 0 l1+l2]);
    axis equal;
    axis square;
    % ====== Plot locations for the training data
    load invkine.mat
    tmp = line(invkine1(:, 1), invkine1(:, 2));
    set(tmp, 'Marker', 'x', 'LineStyle', 'none');
    xlabel('X'); ylabel('Y');
    % ====== set 'userdata'
    tmp = [arm1H arm2H dotH desired_trajH actual_trajH -1 -1 -1 -1 -
1];
    set(InvKineFigH, 'userdata', [get(InvKineFigH, 'userdata');
tmp]);
elseif strcmp(action, 'single_loop'),
    % ====== get animation objects
    tmp = get(InvKineFigH, 'userdata');
    arm1H = tmp(3, 1);
    arm2H = tmp(3, 2);
```

```
    dotH = tmp(3, 3);
    desired_trajH = tmp(3, 4);
    actual_trajH = tmp(3, 5);
    l1 = get(arm1H, 'userdata');
    l2 = get(arm2H, 'userdata');
    countH = tmp(1, 6);
    % ====== get FIS handle
    fisH1 = tmp(3, 6);
    fisH2 = tmp(3, 7);
    % ====== get desired position
    desired_traj =
get(desired_trajH,'xdata')+j*get(desired_trajH,'ydata');
    desired_pos = desired_traj(rem(InvKineCount,
length(desired_traj))+1);
    x = real(desired_pos);
    y = imag(desired_pos);
    % ====== evaluate FIS to get joint angles
    theta1 = evalfis([x, y], InvKineFisMat1);
    theta2 = evalfis([x, y], InvKineFisMat2);
    % ====== count operation
    set(countH, 'string', int2str(InvKineCount));
    InvKineCount = InvKineCount+1;
    % ====== update animation objects
    set(dotH, 'xdata', x, 'ydata', y);
    end1 = l1*exp(j*theta1);
    end2 = end1 + l2*exp(j*(theta1+theta2));
    set(arm1H, 'xdata', [0 real(end1)], 'ydata', [0 imag(end1)]);
    set(arm2H, 'xdata', [real(end1) real(end2)], ...
        'ydata', [imag(end1) imag(end2)]);
    tmp_x = get(actual_trajH, 'xdata');
```

```
        tmp_y = get(actual_trajH, 'ydata');
        set(actual_trajH, 'xdata', [tmp_x(2), real(end2)], ...
            'ydata', [tmp_y(2), imag(end2)]);
        drawnow;
    elseif strcmp(action, 'main_loop'),
        InvKineAnimRunning = 1;
        % ====== get animation objects
        tmp = get(InvKineFigH, 'userdata');
        % ====== change visibility of GUI's
        set(tmp(1, 1), 'visible', 'off');
        set(tmp(1, 4:5), 'visible', 'off');
        set(tmp(1, 2:3), 'visible', 'on');
        % ====== looping
        while 1 & ~InvKineAnimClose
            if ~InvKineAnimRunning | InvKineAnimPause,
                break;
            end
            eval([mfilename, '(''single_loop'')']);
        end
        % ====== shut down
        if InvKineAnimClose,
            delete(InvKineFigH);
        end
    elseif strcmp(action, 'load_fis_mat'),
        tmp = get(InvKineFigH, 'userdata');
        % ====== Read FIS matrices if necessary
        if tmp(3, 6) < 0,    % FIS not been build
            % ====== read FIS matrix
            InvKineFisMat1 = readfis('invkine1');
```

```
        InvKineFisMat2 = readfis('invkine2');
        % ====== change flag
        tmp(3, 6) = 1;
        set(InvKineFigH, 'userdata', tmp);
    end
elseif strcmp(action, 'mouse_action1'),
    % mouse action when button is first pushed down
    tmp = get(InvKineFigH, 'userdata');
    desired_trajH = tmp(3, 4);
    curr_info = get(gca, 'CurrentPoint');
    InvKineCurrPt = curr_info(1, 1) + j*curr_info(1,2);
    now_ball_x = get(desired_trajH, 'xdata');
    now_ball_y = get(desired_trajH, 'ydata');
    now_ball = now_ball_x + j*now_ball_y;
    InvKineInsideEllipse = inside(InvKineCurrPt, now_ball.');
elseif strcmp(action, 'mouse_action2'),
    % mouse actions after the mouse is pushed down and dragged
    if InvKineInsideEllipse,
        tmp = get(InvKineFigH, 'userdata');
        desired_trajH = tmp(3, 4);
        prev_pt = InvKineCurrPt;
        curr_info = get(gca, 'CurrentPoint');
        InvKineCurrPt = curr_info(1,1) + j*curr_info(1,2);
        displace = InvKineCurrPt - prev_pt;
        old_ball_x = get(desired_trajH, 'xdata');
        old_ball_y = get(desired_trajH, 'ydata');
        new_ball = old_ball_x + j*old_ball_y + displace;
        set(desired_trajH, 'xdata', real(new_ball));
        set(desired_trajH, 'ydata', imag(new_ball));
```

```
    end
elseif strcmp(action, 'mouse_action3'),
    % mouse action when button is released
    eval([mfilename, '(''mouse_action2'')']);

elseif strcmp(action, 'stop_anim'),
    tmp = get(InvKineFigH, 'userdata');
    set(tmp(1, 1), 'visible', 'on');
    set(tmp(1, 2:5), 'visible', 'off');
    InvKineAnimRunning = 0;
    InvKineAnimPause = 0;
elseif strcmp(action, 'pause_anim'),
    tmp = get(InvKineFigH, 'userdata');
    set(tmp(1, 3), 'visible', 'off');
    set(tmp(1, 4:5), 'visible', 'on');
    InvKineAnimRunning = 0;
    InvKineAnimClose = 0;
    InvKineAnimPause = 1;
elseif strcmp(action, 'step_anim'),
    InvKineAnimStepping = 1;
    eval([mfilename, '(''single_loop'')']);
elseif strcmp(action, 'continue_anim'),
    tmp = get(InvKineFigH, 'userdata');
    set(tmp(1, 3), 'visible', 'on');
    set(tmp(1, 4:5), 'visible', 'off');
    InvKineAnimRunning = 1;
    InvKineAnimPause = 0;
    InvKineAnimClose = 0;
    InvKineAnimStepping = 0;
```

```
    eval([mfilename, '(''set_constant'')']);
    eval([mfilename, '(''main_loop'')']);
```

A06.2.3.Defuzificação do Controle em Lógica Nebulosa (ou Fuzzy)

```
function defuzzdm
%DEFUZZDM Defuzzification methods.
%   DEFUZZDM displays five defuzzification methods supported in the
%   Fuzzy Logic Toolbox.
%
%       See also DEFUZZ.

%   Roger Jang, 10-28-93, 9-29-94.
%   Copyright (c) 1994-98 by The MathWorks, Inc.
%   $Revision: 1.4 $  $Date: 1997/12/01 21:44:10 $

FigTitle = 'Defuzzification Methods';
fig = findobj(0,'Name',FigTitle);

if isempty(fig),
    fig = figure('Unit','pixel',...
        'Name',FigTitle,...
        'NumberTitle','off');
%%%%    set(0,'Current',fig)
end

x = -10:0.1:10;
offset = 0.03;
```

```
mf1 = trapmf(x, [-10, -8, -2, 2]);

mf2 = trapmf(x, [-5, -3, 2, 4]);

mf3 = trapmf(x, [2, 3, 8, 9]);

mf1 = max(0.5*mf2, max(0.9*mf1,0.1*mf3));


lineColor=[0 0 1];

lineStyle=':';

dotColor=[0 0 1];

set(fig,'DefaultTextFontWeight','bold','DefaultTextColor','k')

axColor=[.4 .4 .4];

set(fig,'Color',[1 1
1],'DefaultAxesXColor',axColor,'DefaultAxesYColor',axColor)


subplot(211);

plot(x, mf1,'LineWidth',3);

patch(x,mf1,'y')

set(gca,'XLim',[min(x) max(x)],'YLim',[0 1.2],'Color',[.9 .9 .9]);


hold on;


x1 =  defuzz(x, mf1, 'centroid');

plot([x1 x1], [0 1.2], 'Color',lineColor,'LineStyle',lineStyle);

plot(x1, 0.8, 'Color',dotColor,'Marker','.','MarkerSize',20);

x2 =  defuzz(x, mf1, 'bisector');

plot([x2 x2], [0 1.2], 'Color',lineColor,'LineStyle',lineStyle);

plot(x2, 0.6, 'Color',dotColor,'Marker','.','MarkerSize',20);

x3 =  defuzz(x, mf1, 'mom');

plot([x3 x3], [0 1.2], 'Color',lineColor,'LineStyle',lineStyle);

plot(x3, 0.4, 'Color',dotColor,'Marker','.','MarkerSize',20);
```

```
x4 =  defuzz(x, mf1, 'som');
plot([x4 x4], [0 1.2], 'Color',lineColor,'LineStyle',lineStyle);
plot(x4, 0.2, 'Color',dotColor,'Marker','.','MarkerSize',20);
x5 =  defuzz(x, mf1, 'lom');
plot([x5 x5], [0 1.2], 'Color',lineColor,'LineStyle',lineStyle);
plot(x5, 0.2, 'Color',dotColor,'Marker','.','MarkerSize',20);

text(x1, 0.8-offset, 'centroid', 'hor', 'center', 'ver', 'top');
text(x2, 0.6-offset, 'bisector', 'hor', 'center', 'ver', 'top');
text(x3, 0.4-offset, 'mom', 'hor', 'center', 'ver', 'top');
text(x4, 0.2-offset, 'som', 'hor', 'center', 'ver', 'top');
text(x5, 0.2-offset, 'lom', 'hor', 'center', 'ver', 'top');
hold off

subplot(212);
mf1=fliplr(mf1);
plot(x, mf1,'LineWidth',3);
patch(x,mf1,'y')
set(gca,'XLim',[min(x) max(x)],'YLim',[0 1.2],'Color',[.9 .9 .9]);

hold on;

x1 =  defuzz(x, mf1, 'centroid');
plot([x1 x1], [0 1.2], 'Color',lineColor,'LineStyle',lineStyle);
plot(x1, 0.8, 'Color',dotColor,'Marker','.','MarkerSize',20);
x2 =  defuzz(x, mf1, 'bisector');
plot([x2 x2], [0 1.2], 'Color',lineColor,'LineStyle',lineStyle);
plot(x2, 0.6, 'Color',dotColor,'Marker','.','MarkerSize',20);
x3 =  defuzz(x, mf1, 'mom');
```

```
plot([x3 x3], [0 1.2], 'Color',lineColor,'LineStyle',lineStyle);
plot(x3, 0.4, 'Color',dotColor,'Marker','.','MarkerSize',20);
x4 =  defuzz(x, mf1, 'som');
plot([x4 x4], [0 1.2], 'Color',lineColor,'LineStyle',lineStyle);
plot(x4, 0.2, 'Color',dotColor,'Marker','.','MarkerSize',20);
x5 =  defuzz(x, mf1, 'lom');
plot([x5 x5], [0 1.2], 'Color',lineColor,'LineStyle',lineStyle);
plot(x5, 0.2, 'Color',dotColor,'Marker','.','MarkerSize',20);

text(x1, 0.8-offset, 'centroid', 'hor', 'center', 'ver', 'top');
text(x2, 0.6-offset, 'bisector', 'hor', 'center', 'ver', 'top');
text(x3, 0.4-offset, 'mom', 'hor', 'center', 'ver', 'top');
text(x4, 0.2-offset, 'som', 'hor', 'center', 'ver', 'top');
text(x5, 0.2-offset, 'lom', 'hor', 'center', 'ver', 'top');
hold off

function out = defuzz(x, mf, type)
% DEFUZZ Defuzzify membership function.
%   Synopsis
%   out = defuzz(x,mf,type)
%   Description
%   defuzz(x,mf,type) returns a defuzzified value out, of a 
membership function
%   mf positioned at associated variable value x, using one of 
several
%   defuzzification strategies, according to the argument, type. The 
variable
%   type can be one of the following.
%   centroid: centroid of area method.
%   bisector: bisector of area method.
```

```
%   mom: mean of maximum method.
%   som: smallest of maximum method.
%   lom: largest of maximum method.
%   If type is not one of the above, it is assumed to be a user-
defined
%   function. x and mf are passed to this function to generate the
defuzzified
%   output.
%   Examples
%   x = -10:0.1:10;
%   mf = trapmf(x,[-10 -8 -4 7]);
%   xx = defuzz(x,mf,'centroid');
%
%   Try DEFUZZDM for more examples.
%   Roger Jang, 6-28-93 ,10-5-93, 9-29-94.
%   Copyright (c) 1994-98 by The MathWorks, Inc.
%   $Revision: 1.7 $  $Date: 1997/12/01 21:44:46 $

x = x(:);
mf = mf(:);
if length(x) ~= length(mf),
    error('Sizes mismatch!');
end
data_n = length(x);

if strcmp(type, 'centroid'),
    total_area = sum(mf);
    if total_area == 0,
        error('Total area is zero in centroid defuzzification!');
    end
```

```
    out = sum(mf.*x)/total_area;
    return;
elseif strcmp(type, 'bisector'),
        total_area = sum(mf);
    if total_area == 0,
        error('Total area is zero in bisector defuzzification!');
    end
        tmp = 0;
        for k=1:data_n,
                tmp = tmp + mf(k);
                if tmp >= total_area/2,
                        break;
                end
        end
    out = x(k);
    return;
elseif strcmp(type, 'mom'),
        out = mean(x(find(mf==max(mf))));
    return;
elseif strcmp(type, 'som'),
        tmp = x(find(mf == max(mf)));
        [junk, which] = min(abs(tmp));
    out = tmp(which);
    return;
elseif strcmp(type, 'lom'),
        tmp = x(find(mf == max(mf)));
        [junk, which] = max(abs(tmp));
    out = tmp(which);
    return;
```

```
else
    % defuzzification type is unknown
    % We assume it is user-defined and evaluate it here
    evalStr=[type '(x, mf)'];
    out = eval(evalStr);
    return;
end
```